INTERNE BERICHTE

Carl von Ossietzky Universität Oldenburg

Fachbereich Informatik

Endbericht der Projektgruppe

Werkzeuge für Digitale Bibliotheken

J. Eschke, M. Höft, M. Hülsmann, M. Klein, O. Klein, M. Malachinski, T. Ottenhues, D. Pawlowski, A. Rolfs, V. Weber, W. Willms, A. Wunsch, D. Boles, C. Haber

Bericht IS xx

Oktober 2000

Inhaltsverzeichnis

1 Einleitung **1**

1.1 Inhalte und Ziele der Projektgruppe 1

1.2 Digitale Bibliotheken 3

1.3 Electronic Commerce 3

1.4 Digital Commerce 4

1.5 Aufbau der Arbeit 5

2 Das DDS-System **7**

2.1 Motivation 7

2.2 Anforderungen 8

2.3 Architektur 9

2.3.1 Datenbank-Komponente 10

2.3.2 Payment-Komponente 10

2.3.3 Dokumentenschutz-Komponente 10

2.3.4 Authoring-Komponente 10

2.3.5 Web-Komponente 11

2.4 Realisierung 11

2.5 Installationsanleitung 12

3 Datenbank-Komponente **15**

3.1 Datenbank-Entwurf 15

3.1.1 ER-Schemata 15

3.1.2 DB-Tabellendefinitionen 19

3.2 Datenbank-Schnittstellen 28

3.2.1 Die Schnittstellen von DB_Access 28

3.2.2 Die Schnittstellen für das DDS-System 28
3.2.3 Die Schnittstellen für den Payment-Handler 40
3.2.4 Ausnahmen- und Fehlerklassen 42
3.3 Realisierung 45
3.3.1 Zeitliche Vorgehensweise bei der Realisierung 45
3.3.2 Aufbau der DDS-DB-Komponente 46
3.4 DB-Optimierungsaspekte 47

4 Payment - Komponente 49
4.1 Entwurf 49
4.2 Schnittstellen 50
4.2.1 Allgemeines 50
4.2.2 PaymentHandlerInterface 50
4.2.3 Brand 53
4.2.4 InvoiceResult 54
4.2.5 Login 54
4.2.6 Transaction 55
4.3 Architektur und Implementierung des Payment-Systems 55
4.3.1 Klassen des Payment-Systems 56
4.3.2 Implementierung der Clients für die Zahlungsverfahren 59
4.3.3 Erweiterbarkeit des PaymentHandlers 60
4.4 Benutzungshandbuch 61
4.4.1 Einrichtung und Betrieb der Payment-Komponente 61
4.4.2 Das Administrations-Tool zur Transaktions-Verwaltung 64

5 Dokumentschutz-Komponente 67
5.1 Motivation 67
5.2 Das Lizenz-Konzept 67
5.2.1 Eine Aufzählung von möglichen Sicherheitstechniken , die auf das Lizenz-Konzept aufsetzen bzw. es ersetzen 68
5.3 Analyse und Problembetrachtung der unterschiedlichen Sicherheitstechniken . . . 69
5.4 Entwurf 71
5.4.1 Anfrage über Socket-Verbindung 71

5.4.2 Der first_click-Request 76
5.4.3 Der Standard-Request 77
5.5 Realisierung der gewählten Sicherheitstechniken 77
5.5.1 Implementation des Server-Plugins 78
5.5.2 Implementation des Security-Servlets 79
5.5.3 Kommunikation zwischen Server-Plugin und Security-Servlet 80
5.6 Schnittstellenbeschreibung 80
5.7 Erweiterbarkeit der vorhandenen Sicherheitstechniken 82
5.8 Das Problem bei der Umbenennung der Dateinamen 82

6 Authoring - Komponente 85
6.1 Entwurf 85
6.1.1 Designentscheidung 85
6.1.2 Entwurf der Benutzer-Schnittstelle 88
6.2 Implementierung 99
6.2.1 Schnittstellen-Dokumentation 99
6.2.2 Problemlösungen 101
6.3 Benutzungshandbuch 104
6.3.1 Benutzungshandbuch des Authoring-Tools 104
6.3.2 Erläuterung der einzelnen Fenster 104
6.3.3 Vorgehensweise bei den wichtigsten Arbeitsschritten 114

7 DDS - Web - Komponente 117
7.1 Einleitung 117
7.2 Entwurf und Ipmlementierung 118
7.2.1 Der Meilensteinplan 118
7.2.2 Der Entwurf und die Implementierung 118
7.2.3 Die Architektur des Webshop's 121
7.3 Benutzerhandbuch 124
7.3.1 Erklärungen für den Betreiber 125
7.3.2 Handbuch für den Benutzer 125
7.3.3 Die Interaktionsmöglichkeiten des Benutzers 145
7.4 Klassenstruckturen 149

8 Bewertung und Ausblick 151

Abbildungsverzeichnis

2.1 Die Architektur des DDS-Systems 9

3.1 Entity-Relationship Diagramm für die DDS-Einstellungen 15
3.2 Entity-Relationship Diagramm für den DDS 16
3.3 Entity-Relationship Diagramm für den Payment-Manager 18
3.4 webshop.dbif - Paketstruktur 46

4.1 Klassenhierarchie des Payment-Systems 56
4.2 Screenshot des Administrations-Tools 64

5.1 Ist die Session-ID jemals gültig gewesen? 71
5.2 Ist die Session-ID noch gültig bzgl. der 30-Minuten-Grenze? 72
5.3 Ist der first_click gesetzt? 73
5.4 Ist die Lizenz schon bezahlt? 74
5.5 Ist die Lizenz noch gültig? 75
5.6 Der first_click-Request 76
5.7 Der Standard-Request 77

6.1 Das allgemeine Klassenmodell des GUI 87
6.2 Das spezielle Klassenmodell des GUI 88
6.3 Shop-Einstellungen ändern 89
6.4 Neues Dokument 90
6.5 Dokument löschen 90
6.6 Dokument ändern 91
6.7 Authoring-Tool: Hauptfenster mit Objekten 92
6.8 Authoring-Tool: Detail-Informationen zu einem Objekt 92
6.9 Authoring-Tool: Angebote zu einem Objekt 93

6.10 Authoring-Tool: Autoren zu einem Objekt 93
6.11 Authoring-Tool: URLs zu einem Objekt 94
6.12 Authoring-Tool: Auswahl neuer URLs (Std. Filechooser) 94
6.13 Authoring-Tool: Autoren-Tabelle in der Datenbank 95
6.14 Authoring-Tool: Verlage-Tabelle in der Datenbank 95
6.15 Authoring-Tool: Formate-Tabelle in der Datenbank 95
6.16 Authoring-Tool: Sprachen-Tabelle in der Datenbank 96
6.17 Authoring-Tool: Typen-Tabelle in der Datenbank 96
6.18 Authoring-Tool: Allgemeine Shop-Einstellungen 97
6.19 Authoring-Tool: Internet-Einstellungen des Shops 97
6.20 Authoring-Tool: Payment-Einstellungen des Shops 97
6.21 Authoring-Tool: Auswahl eines Layouts 98
6.22 Authoring-Tool: Detailansicht eines Layouts 98
6.23 Das Hauptfenster des Authoring-Tools 105
6.24 Die generellen Shop-Einstellungen 107
6.25 Die Internet Shop-Einstellungen 108
6.26 Die Bezahlungs Shop-Einstellungen 108
6.27 Ein Datenbankfenster des Authoring-Tools 109
6.28 Das Details-Fenster des Authoring-Tools 109
6.29 Auswahl der Artikel-Inhalte 111
6.30 Das Autoren-Fenster des Authoring-Tools 111
6.31 Das Angebote-Fenster des Authoring-Tools 112
6.32 Der Dateimanager 113

7.1 Die Klassenstruktur der Servlets 121
7.2 Die Klassenstruktur der ShopPages 122
7.3 Das Layout 1 126
7.4 Die Navigation durch die Dokumentkategorien im Layout 1 127
7.5 Die Navigationsleiste von Layout 1 127
7.6 Das Layout 2 129
7.7 Die Navigation durch die Dokumentkategorien im Layout 2 130
7.8 Das Layout 3 131

7.9 Die Navigation im Layout 3 132
7.10 Das Login-Fenster 134
7.11 Der Warenkorb 135
7.12 Das Benutzerdaten-Fenster 137
7.13 Das Hilfe-Fenster 138
7.14 Das Suchen-Fenster 138
7.15 Das ShopInfo - Fenster 140
7.16 Das Lizenz-Anzeige 141
7.17 Die Langinfo zu einem Dokument 142
7.18 Das Kundenkonto-Fenster 142
7.19 Die Dokumentanzeige 143
7.20 Die Interfaces 149
7.21 Die Klasssen die FormSelector implementieren 149
7.22 sonstige Klasssen 149
7.23 Die Klasssen die FormContent implementieren 150

Kapitel 1

Einleitung

1.1 Inhalte und Ziele der Projektgruppe

Dieser Bericht fasst die Ergebnisse der Arbeit der Projektgruppe tooLib (Werkzeuge für Digitale Bibliotheken) zusammen, die im Wintersemester 1999/2000 und im Sommersemester 2000 im Fachbereich Informatik der Universität Oldenburg stattfand.
Der Ablauf und die Inhalte der Projektgruppe lassen sich in zwei Abschnitte gleidern.

Phase 1 Als erstes Teilziel innerhalb der Projektgruppe sollten die Anforderungen, Eigenschaften, Funktionsweise, Aufbau sowie die Probleme mit der Entwicklung von Digitalen Bibliotheken näher kennen gelernt werden. Damit die Teilnehmer der Projektgruppe in kleinerem Umfang praktische Erfahrungen auf diesem Gebiet sammeln konnten, beschäftigten sie sich mit der prototypischen Entwicklung eigener Digitaler Bibliotheken mit klar abgegrenzten und eingeschränkten Inhalten in einem festgelegten thematischen Kontext. Insbesondere sollten hier auch erste Erfahrungen mit Softwarewerkzeugen gesammelt werden, die die Entwicklungsarbeit unterstützen können.
Genauer ging es um die folgenden Themen:

Das Sub-Projekt Sport-DL Das Sub-Projekt, das unter dem Titel Sport-DL lief, beschäftigte sich mit der Entwicklung einer Digitalen Bibliothek im sportwissenschaftlichen Anwendungsbereich. Für das Projekt Sport-DL gab es mit dem Institut für Sportwissenschaft der Universität Saarland einen Auftraggeber, der unter der Überschrift *Web-based Publishing* einen Anforderungskatalog bereitstellte, das Projekt selbst aber nicht weiter begleitete. Das Institut der Uni-Saarland startete 1998 das europäische Pilot Projekt ITES (Information Technologies in European Sport and Sport Science), das sich aus vier Teilen zusammensetzt, die sich mit jeweils unterschiedliche Zielen internetbasierte virtuelle Kommunikationsmöglichkeiten zunutze machen wollen. Das Ziel der Sport-DL war es, im Internet ein Angebot zu schaffen, das die Möglichkeit bietet, sportwissenschaftliche Fachbeiträge aus dem Bereich Motor Control and Learning zu publizieren sowie umfassende Recherchemöglichkeiten zur Verfügung zu stellen.

Das Sub-Projekt Bean-DL Die Aufgabe der BeanDL-Gruppe war es, auf Grundlage des Bean Development Kit 1.1 und des Java Development Kit 1.2, eine neue BeanBox zu erstellen, diese neue BeanBox bekam von den Mitgliedern der BeanDL-Gruppe

den Namen NewBeanBox. Die NewBeanBox sollte der Aufgabenstellung entsprechend angepasst werden. Diese Aufgabenstellung umfasste vor allem die Umwandlung des BDK in eine verteilte digitale Bibliothek auf Grundlage von JavaBeans. Das heißt das bestehende BDK musste so umgewandelt werden, dass es sowohl JavaBeans aus dem Internet als auch JavaBeans, die lokal auf der Festplatte liegen, verwenden kann. Diese Aufgabe erforderte eine Anpassung der ToolBox, sowie die Implementierung eines Servers, über den die NewBeanBox auf die JavaBeans im Internet Zugang erhält und eine WWW-Schnittstelle für die KleinAnbieter einzelner JavaBeans. Mit der NewBeanBox soll es möglich sein, JavaBeans über das Internet von verschiedenen Servern zu verwenden aber auch weiterhin lokale JavaBeans zu benutzen. Eine der weiteren Aufgaben war die Implementierung einer Suche auf den angebotenen JavaBeans. Das heißt man kann die JavaBeans, die einem lokal oder über einen der Server zur Verfügung stehen, nach verschiedenen Kriterien durchsuchen. Der Schwerpunkt dieser Gruppe lag auf der Benutzerseite nicht auf der Anbieterseite, das heißt die NewBeanBox wurde hauptsächlich für den Anwender optimiert und nur zum geringen Teil für den Anbieter.

Das Sub-Projekt Framework Im Rahmen der gestellten Aufgabe sollte ein allgemeines Modell entwickelt werden, das das Angebot, die Bestellung, die Abrechnung und die Bezahlung digitaler Objekte unterstützt. Die konkrete Umsetzung des entwickelten Modells sollte anschließend durch eine prototypische Implementierung erfolgen.
Da es sich um ein allgemeines Modell handelt, sollen bezüglich der Art der digitalen Objekte keinerlei Einschränkungen gemacht werden. Als typische digitale Objekte, die angeboten werden können, seien jedoch Texte, Bücher, Musik, Software etc. genannt. Durch das Framework sollen weder die Zustellung der Objekte zum Käufer noch deren Benutzung durch den Käufer berücksichtigt bzw. implementiert werden.
Für das Angebot digitaler Objekte soll der Anbieter beispielsweise in der Lage sein, den angebotenen Objekten individuelle Preise zu geben sowie Rabatte, Sonderangebote und Preisnachlässe zuzuweisen. Letztere können z.B. von der Nutzungsform und dem Käufer bzw. der Käufergruppe abhängig sein. Bei der Bestellung der digitalen Objekte soll der Käufer zwischen verschiedenen Nutzungsformen der Ware (Gleitlizenz, Dauerlizenz, Campuslizenz etc.) wählen können. Für erworbenene Objekte erhält er dann die entsprechende Lizenz. Die Bezahlung der Objekte soll über verschiedene elektronische Zahlungsverfahren wie Ecash oder SET möglich sein. Insbesondere soll die Verwendung guthabenbasierter Verfahren durch die Verwaltung von Zahlungskonten durch das Framework unterstützt werden.

Grundlegende Aufgabe des zu entwickelnden Frameworks ist es, alle Daten, die für Angebot, Bestellung und Abrechnung digitaler Objekte erforderlich sind, sowie die Beziehungen zwischen diesen Daten zu verwalten. Dies umfasst unter anderem die Verwaltung von Waren, Angeboten zu Waren, Käufern und Käufergruppen, Konten, Rabatten, Bestellungen und erworbenen Lizenzen.
Weitere Aufgabe ist es, die Funktionstauglichkeit des entwickelten Modells durch eine beispielhafte Implementierung zu demonstrieren. Dazu sollen zwei Programme entwickelt werden: je eines für den Anbieter digitaler Waren und eines für den potentiellen Käufer der Waren.

Nähres zu den drei Sub-Projekten kann im Zwischenbrericht nachgelesen werden.

Phase 2 Der zweite Abschnitt des Projekts sollte als Endprodukt ein EShop-System für den Aufbau, Einsatz und zur Verwaltung von EShops für den Handel mit digitale Dokumente aufweisen, mit denen der kostenpflichtige Zugriff auf digitale Informationen realisiert wird. Im vorliegenden Bericht soll das Tool als *DDS-System (Digital Documents Shop)* bezeichnet werden. Einleitend soll beschrieben werden, inwieweit sich digitale Bibliotheken und E-Commerce im weitesten Sinne voneinander abgrenzen bzw. wo sich mögliche Schnittpunkte ergeben können.

1.2 Digitale Bibliotheken

Die Hauptbestandteile digitaler Bibliotheken sind organisierte Sammlungen digitaler Informationen, die über das Internet angesprochen werden können. Im Gegensatz zu traditionellen Bibliotheken kann eine digitale Bibliothek nicht nur materielle Dokumente sondern auch immaterielle Dokumente beinhalten. Die Aufgaben der digitalen Bibliothek ändern sich nicht. Es können aber eine größere Anzahl an Dokumenten aufgenommen werden, da die digitale Speicherung weniger räumlichen Platz verbraucht. Auch kann nur der Link zu einem Dokument gespeichert werden, so dass nicht jede Bibliothek alle Dokumente vor Ort haben muss, sondern nur einige wenige Exemplare. Dabei beschränken sich digitale Bibliotheken nicht nur auf Textdokumente. Sie können außerden alle denkbaren digiteln Formate aufnehmen und dem Anwender zugänglich machen. Das können Videos, Musikstücke, Bilder oder komlette Softwareprogramme sein.

Soll das Angebot für den Nutzer kostenpflichtig zur Verfügung gestellt werden, gilt es entsprechende Lizenzmodelle, Angebotsformen, Bestellformen, Abrechnungsverfahren, Zahlungsverfahren und Auslieferungsverfahren zu integrieren. Der Kreis der Akteure erweitert sich hierbei im Gegensatz zu herkömmlichen Bibliotheken mit mindestens dem Verleiher und dem Nutzer um Administratoren für die IT-Komponenten (Datenbank-, WWW-Administratur etc.) sowie die IT-Komponenten selbst. Bei kostenpflichtigen digitalen Bibliotheken kommen noch Akteure hinzu, die für die Abwicklung der Zahlungen zuständig sind.

1.3 Electronic Commerce

Als Electronic Commerce wird der Handel mit Gütern über Kommunikationsnetze, insbesondere das Internet bezeichnet. Bei den Gütern handelt es sich heutzutage vorwiegend um materielle Güter wie Bücher, CDs oder andere Gebrauchsgegenstände. Der Verkauf dieser Güter ist über spezielle WWW-Sites, sogenannte eShops (Electronic Shops) im WWW möglich. Für die Erstellung und Pflege solcher Shops gibt es inzwischen zahlreiche Software-Systeme, sogenannte eShop-Systeme oder Commerce Server (InterShop, Openshop, Internolix, ...).

eShops sind spezielle WWW-Sites, die den Kauf von Produkten über das WWW ermöglichen. Die Funktionalität von eShops basiert eigentlich immer auf demselben Muster, das traditionellen Einkaufsvorgängen nachgebildet ist: Kunden können entweder über einen Katalog im Produktbestand navigieren oder gezielt nach Produkten suchen. Aus den Trefferlisten lassen sich Produkte auswählen und ausführlichere Beschreibungen, häufig mit Fotos oder Filmen versehen, abrufen. Produkte, die der Kunde erwerben möchte, können in einen virtuellen Warenkorb abgelegt werden. Hat der Kunde seine Auswahl abgeschlossen, begibt er sich zur „Kasse“. Hier

sind die Zahlungsformalitäten zu erledigen. Anschließend werden die erworbenen Produkte im allgemeinen per Post ausgeliefert.

eShop-Systeme sind Software-Systeme, die den Aufbau, die Verwaltung und den Einsatz von eShops unterstützen. Für den Händler werden zumeist graphisch-interaktive Tools zur Verfügung gestellt, die es erlauben, auch ohne Programmierkenntnisse einen eShop einzurichten und zu betreiben. Die Daten werden in einer Datenbank abgespeichert. Der Kundenzugriff wird über dynamische WWW-Seiten realisiert, wobei die Daten aus der Datenbank abgerufen werden.

1.4 Digital Commerce

Als digital Commerce wird der Handel mit digitalen Produkten (Texte, Bilder, Videos, Musikstücke, ...) bzw. Informationen (Börsenkurse, Verkehrsmeldungen, ...) bezeichnet. Gebräuchlich ist auch der Ausdruck „Information Commerce“.

Als „iShops“ (Information Shops) bezeichnen wir spezielle eShops, über die keine materiellen sondern digitale Produkte bzw. Informationen über das WWW erworben werden können. Als digitale Produkte können vor allem textuelle Informationen, wie hypermediale Dokumente, aber auch Musikstücke, Videos oder Software kommerziell vertrieben werden.

Ein weitreichernder Unterschied zwischen iShops und eShops besteht in der Möglichkeit, beim Verkauf von digitalen Produkten zusätzliche Angebotsformen bzw. Geschäftsmodelle zu unterstützen. In kommerziellen digitalen Bibliotheken wird heutzutage nahezu ausschließlich die Subskription angeboten: Der Kunde zahlt einen bestimmten Betrag und darf dann für eine gewisse Zeitspanne auf den gesamten Dokumentenbestand zugreifen. Eine interessante Alternative hierzu ist der Erwerb von Zugriffslizenzen (pay-per-view) oder Zeitlizenzen: Bei den Zugriffslizenzen wird für jeden Zugriff auf ein digitales Produkt ein bestimmter (im allgemeinen sehr geringer) Geldbetrag fällig. Zum Bezahlen kann hier bspw. auf elektronisches Geld (Mikropayments) zurückgegriffen werden. Als Zeitlizenzen werden Lizenzen bezeichnet, mit denen der Kunde eine bestimmte Zeitspanne lang auf ein ausgewähltes Produkt zugreifen kann. Weitere denkbare Angebotsformen, die durch iShops in besonderer Weise unterstützt werden können, entstammen dem Softwarebereich: Shareware, kostengünstigere Demoversionen, eingeschränkt nutzbare Versionen (Leseproben) oder die immer beliebter werdende Angebotsform „first-try-then-buy“.

Die Unterstützung institutioneller Nutzergruppen ist ein weiterer Aspekt, für den iShops geradezu prädestiniert sind. Hier ist zum Beispiel der Erwerb von Campuslizenzen denkbar: Ein Gruppenadministrator erwirbt für eine bestimmte Zeitspanne eine Campuslizenz für ein digitales Produkt und alle Gruppenangehörigen können dann auf das Produkt zugreifen. Das Lizenzmodell der Gleitlizenzen ist dem Ausleihverfahren in traditionellen Bibliotheken nachempfunden: Besitzt eine Gruppe n Gleitlizenzen für ein digitales Produkt, können maximal n Gruppenangehörige gleichzeitig auf das Produkt zugreifen. U.U. für Hochschulbibliotheken interessant ist die Einführung und Verwaltung sogenannter Gruppenkonten: Der Gruppenadministrator kann dort einen bestimmten Geldbetrag einzahlen und den Gruppenmitgliedern erlauben, bis zu einem pro Person festgelegten Betrag das Geld für den Erwerb von Einzellizenzen zu nutzen. Auf diese Art und Weise könnten die Bibliotheken sehr flexibel auf konkrete Nutzerbedürfnisse reagieren.

iShop-Systeme dienen, wie die bereits beschriebenen eShop-Systeme, dem Aufbau, der Verwaltung und dem Einsatz von Shops, hier iShops. Sie bieten aber einen im Hinblick auf Lizenzmodelle und Zugriffsmechanismen eine erweiterte Funktionalität.

1.5 Aufbau der Arbeit

Das vorliegende Dokument ist wie folgt gegliedert:

- Kapitel 1 - Einleitende Worte zum Inhalt und den Zielen der Projektgruppe. (dieses Kapitel)
- Kapitel 2 - Motivation, Anforderungen, Architektur, Realisierung und Installation des DDS-Systems.
- Kapitel 3 - Beschreibung der Datenbank-Komponente: DB-Entwurf, -Tabellen, -Schnittstellen, -Realisierung und -Optimierungsaspekte.
- Kapitel 4 - Beschreibung der Payment-Komponente: Entwurf, Architektur, Realisierung, Schnittstellen und Erweiterungsmöglichkeiten des Payment-Händlers.
- Kapitel 5 - Beschreibung des Servers: Motivation, Entwurf, Realisierung, Schnittstellen, Dokumentenschutzmechanismen und deren Erweiterbarkeit sowie Probleme, die zu Einschränkungen des Systems führen.
- Kapitel 6 - Beschreibung des Authoring Tools: Entwurf, Implementierungsaspekte sowie Benuzerhandbuch mit Screendumps des Tools.
- Kapitel 7 - Beschreibung der Web-Komponente: Entwurf, Implementierungsaspekte sowie Benutzerhadbuch mit Screendump der Benutzerschnittstelle.
- Kapitel 8 - Bewertung und Ausblick: Zusammenfassung, Bewertung in Bezug auf die in Kapitel 2 beschriebenen Anforderungen, Ausblick auf mögliche Erweiterungen, Verbesserungen etc.

Kapitel 2

Das DDS-System

Das DDS (Digital-Document-Shop)- System ist ein IShop-System für den Aufbau, die Verwaltung und den Einsatz von IShops mit hypermedialen Dokumenten. Zur Zeit wird der Verkauf von Zeitlizenzen zum Online-Zugriff auf die Dokumente über Standard-WWW-Browser unterstützt. Die Bezahlung der Lizenzen erfolgt über verschiedene vom Anbieter des iShops unterstützte Zahlungsverfahren, wie zum Beispiel ECash, CyberCash, Kreditkarte, Bankeinzug oder Kundenkonto.

2.1 Motivation

Im Gegenteil zum Verkauf von materiellen Gütern in den sogenannten eShops über das WWW ist der Verkauf von digitalen Produkten (wie Texte, Bilder, Videos, Musikstücke,...) oder digitalen Informationen (wie Börsenkurse, Verkehrsmeldungen,...) beziehungsweise der kostenpflichtige Zugriff auf digitale Informationen heutzutage noch sehr wenig verbreitet. Einige Commerce Server unterstützen zwar den Download von digitalen Dokumenten, aber speziell auf den Information Commerce ausgelegte Geschäftsmodelle sind nur unzureichend in die bestehenden EShops integriert.
Bezüglich der Funktionalität unterscheiden sich iShops aus Kundensicht kaum von den eShops. Die einzige Ausnahme ist das Auslieferungsverfahren der gekauften Produkte. Während die materiellen Produkte in den eShops offline ausgeliefert werden - also zum Beispiel als Paket mittels eines Paketbringdienstes - ist die Auslieferung von digitalen Produkten online möglich. Man kann generell zwei unterschiedliche Auslieferungsverfahren für die Auslieferung von digitalen Produkten unterscheiden:

- **direkte Online-Auslieferung:** Das Produkt wird entweder zum Download bereitgestellt oder via eMail an den Kunden verschickt.
- **indirekte Online-Auslieferung:** Das Produkt verbleibt beim Anbieter, der einen Online-Zugriff zumeist über WWW-Browser ermöglicht (sogenannte Online-Lizenzen).

Aus Anbietersicht ist die zweite Auslieferungsform interessanter, da so ein besserer Informationsschutz - beispielsweise was die Verletzung von Copyright und Urheberrecht angeht - gewährleistet werden kann. Hierzu sind natürlich entsprechende Schutzmechanismen in den

iShop zu integrieren bzw. auf dem Server des Anbieters zu installieren.

Im Rahmen unserer Projektgruppe haben wir uns mit Digitalen Bibliotheken beschäftigt und so lag es nahe, zum einen ein Tool zu entwickeln, welches einen Shop-Betreiber bei der Erstellung seines Shops unterstützt und zum anderen auch ein Geschäftsmodell für den Information Commerce in einen solchen iShop integriert.
Das Ziel sollte also sein, ein DDS-System zu entwickeln, welches eine Integration von kostenpflichtigen Digitalen Bibliotheken und E-Commerce realisiert.

2.2 Anforderungen

Das DDS-System sollte es einem Anbieter ermöglichen, fast ohne Programmierkenntnisse oder Kenntnisse in HTML einen eigenen iShop aufzubauen und zu betreiben. Dieser iShop sollte die Möglichkeit bieten, Online-Zeit-Lizenzen zu erwerben und diese mit verschiedensten Zahlungsmethoden zu bezahlen.
Im einzelnen haben wir die folgenden Anforderungen an das zu entwickelnde DDS-System gestellt, welches sich aus fünf verschiedenen Modulen zusammen setzen sollte:

- Ein Tool sollte den Shopbetreiber bei der Erstellung seines iShops untersstützen. Dieses sogenannte Authoring-Tool sollte folgende Anforderungen erfüllen:
  - allgemeine Shopeinstellungen (wie z.B. ShopLayout, Name, Anschrift,...) sollten vorgenommen werden können,
  - die angebotenen Dokumente sollten in Kategorien organisiert werden können (Klassifikation der Dokumente),
  - der Shopbetreiber sollte Kurzbeschreibungen für die Dokumente anfertigen können (z.B. bibliographische Daten, Inhaltsverzeichnis, Leseprobe, Cover,...),
  - angebotene Dokumente dürfen aus (mehreren) URLs bestehen, die frei oder nur mit der entsprechenden Lizenz zugreifbar sein sollten - hierfür sollten die URLs ausgewählt und entsprechend als frei oder nicht frei gekennzeichnet werden können,
  - für die Dokumente sollten Angebote erstellt werden können (Änderung von Lizenzpreisen etc.),
  - der Shopbetreiber sollte die unterstützten Zahlungsverfahren spezifizieren können,
  - die Kundendaten sollten gepflegt werden können,
  - der Shopbetreiber sollte sich eine Übersicht über die verkauften Lizenzen schaffen können,
  - und zur Benutzung sollte eine Hilfefunktion angeboten werden.
- Zur Datenhaltung sollte eine Datenbank via JDBC angebunden werden, so dass quasi jede Datenbank, die JDBC unterstützt, angebunden werden kann. Zum Zugriff auf die Datenbank sollten entsprechenden Schnittstellen für die anderen Module bereitgestellt werden.
- Das DDS-System sollte verschiedene Zahlungsverfahren (wie zum Beispiel eCash, CyberCash, Kreditkarte, ...) für die Bezahlung der Online-Zeit-Lizenzen unterstützen.

- Weiterhin sollte der eigentliche iShop zum Kauf von Online-Zeit-Lizenzen für digitale Dokumente in verschiedenen Layouts erstellt werden, der die folgenden Funktionalitäten bietet:
  - neue Kunden sollten sich registrieren können,
  - bereits registrierte Kunden sollten sich anmelden können,
  - ein Gastzugang sollte möglich sein, so dass erst mal der Shop ausgekundschaftet werden kann, bevor sich ein Kunde registriert,
  - der iShop sollte eine Suchfunktion zur Verfügung stellen (attributierte Suche in Dokumentbeschreibungen),
  - eine Navigation im Produktkatalog (Klassifikation) sollte angeboten werden,
  - die Anzeige von Dokumentbeschreibungen sollte möglich sein,
  - es sollte die Möglichkeit der Lizenzbestellung (Einzelbestellung + Warenkorb) inklusive Bezahlung verfügbar sein,
  - eine Übersicht über die aktuellen Lizenzen sollte angezeigt werden können
  - und der Zugriff auf die hypermedialen Dokumente sollte aus dem iShop heraus möglich sein.
- Durch eine weitere Komponente sollten die Dokumente vor unberechtigtem Zugriff geschützt werden (Session-IDs in Verbindung mit Login und Passwort zur Identifikation).

2.3 Architektur

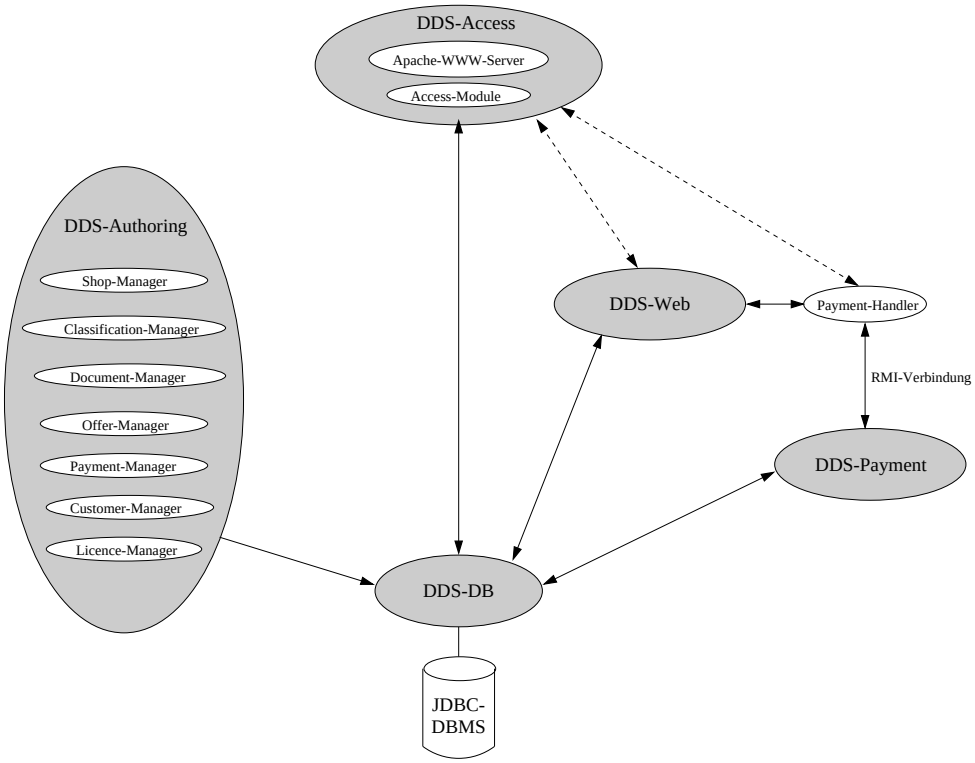

Abbildung 2.1: Die Architektur des DDS-Systems

Die Architektur unseres DDS-Systems wird in Abbildung 2.1 skizziert. Es besteht aus den fünf Komponenten DDS-Authoring (Authoring-Komponente), DDS-DB (Datenbank-Komponente), DDS-Web (Web-Komponente), DDS-Payment (Payment-Komponente) und DDS-Access (Dokumentenschutz-Komponente), die im Folgenden kurz vorgestellt werden sollen. Die Anbindung der DDS-Payment-Komponente an die Komponenten DDS-Access und DDS-Web sowie zum Austausch der möglichen Zahlungsverfahren auch an die Komponente DDS-Authoring geschieht über den RMI-Client des Payment-Handlers. Die Kommunikation zwischen DDS-Payment und DDS-Datenbank erfolgt direkt, da für die Payment-Komponente eigene Tabellen in der Datenbank angelegt werden.

2.3.1 Datenbank-Komponente

Die Daten des iShops werden in einer Datenbank gespeichert, die durch die Komponente DDS-DB gekapselt wird. Der Zugriff auf die Datenbank erfolgt über JDBC (Java Database Connectivity). So können im Prinzip alle Datenbanksysteme verwendet werden, die JDBC unterstützen.

2.3.2 Payment-Komponente

Standardmässig muss sich ein Anbieter selbst um die Abwicklung der elektronischen Zahlungsvorgänge kümmern. Die Mechanismen sind zwar integriert, aber es fallen bspw. bei eCash und CyberCash noch Anmeldungen bei Banken bzw. Kreditunternehmen an. Alternativ kann jedoch auch ein zentraler Payment-Handler angebunden werden, der verschiedene Zahlungsverfahren kapselt und allen Anbietern zur Verfügung gestellt wird.
Die Komponente DDS-Payment stellt zusätzlich noch eine Java-Applikation zur Verwaltung der Bezahlungen zur Verfügung (bspw. für das Eintragen der Zahlungseingänge bei Bankeinzug).

2.3.3 Dokumentenschutz-Komponente

Die Komponente DDS-Access verwaltet die hypermedialen Dokumente und gewährt den autorisierten Zugriff auf die Dokumente. Sie besteht aus dem Apache-WWW-Server sowie einem Apache-erweiternden Modul, welches die Zugriffsschutzmaßnahmen realisiert.

2.3.4 Authoring-Komponente

Die Komponente DDS-Authoring ist als Java-Applikation realisiert worden. Sie unterstützt die Einrichtung und Verwaltung von iShops durch den Einsatz von graphisch-interaktiven Hilfsmitteln. Aus den folgenden Sub-Komponenten setzt sich die Komponente DDS-Authoring zusammen:

- Mit dem Shop-Manager können generelle Einstellungen wie Auswahl des Shop-Layouts, Name des Shops, Name und Anschrift des Shop-Betreibers etc. vorgenommen werden,
- der Classification-Manager ist für die Organisation der einzelnen Dokumente in Kategorien (Klassifikationen) verantwortlich,

- mittels des Document-Managers kann der Anbieter Kurzbeschreibungen der Dokumente (bibliographische Daten, Inhaltsverzeichnis, Leseprobe, Cover, ...) anfertigen bzw. entsprechende Verweise in Form von URLs angeben. Die Dokumente selbst setzen sich aus einer Menge von URLs zusammen, die ebenfalls mit Hilfe des Document-Managers spezifiziert werden können,
- der Offer-Manager ermöglicht das Anlegen von Angeboten,
- über den Payment-Manager lassen sich unterschiedliche Zahlungsverfahren (Bankeinzug, Kreditkarte, eCash, CyberCash und Kundenkonto) selektieren, die vom iShop akzeptiert werden,
- mit dem Customer-Manager können die Kundendaten gepflegt werden
- und mit Hilfe des Licence-Managers kann sich der Anbieter eine Übersicht über die verkauften Lizenzen verschaffen.

2.3.5 Web-Komponente

Der eigentliche iShop aus Kundensicht wird durch die Komponente DDS-Web repräsentiert. Sie ist als Java-Servlet realisiert, das auf Basis der Kundeninteraktionen und der aktuellen Inhalte der Datenbank dynamisch den HTML-Code für die Ausgabe als WWW-Seite generiert. Damit ist der iShop über Standard-WWW-Browser für eine große Masse zugänglich. Die Komponente DDS-Web unterstützt im wesentlichen die folgenden Funktionalitäten:

- Registrierung neuer Kunden,
- Anmeldung bereits registrierter Kunden,
- Gastzugang,
- Suche (attributierte Suche in den Dokumentbeschreibungen),
- Navigation in den Produktkategorien (Klassifikation),
- Anzeige von Dokumentbeschreibungen,
- Lizenzbestellung (Einzelbestellung und Warenkorb) inklusive Bezahlung mittels unterstützter Zahlungsverfahren des iShops,
- Übersicht über die aktuellen Lizenzen eines Nutzers,
- Zugriff auf die hypermedialen Dokumente.

2.4 Realisierung

Das Gesamtprojekt wurde in fünf Gruppen aufgeteilt, die die oben genannten Komponenten DDS-Authoring, DDS-DB, DDS-Payment, DDS-Access und DDS-Web implementieren sollten. Die Schnittstellen zwischen den einzelnen Gruppen wurden als Java-Klassen realisiert, wobei die zentrale Schnittstelle eigentlich für alle Komponenten die Anbindung an die Datenbank

ist. Hierzu wurde die Klasse webshop.dbif.DB_Access von der DDS-DB-Gruppe zur Verfügung gestellt, die die Methoden für den Zugriff auf den Datenbestand der Datenbank realisiert. Der Zugriff auf die Komponente DDS-Payment wurde über die Klasse payment.PaymentHandler (den sogenannten Payment-Handler) realisiert, der als RMI-Client die Kommunikation mit der DDS-Payment-Komponente über eine RMI-Verbindung herstellt.
Die Komponenten DDS-Access und DDS-Web benutzen sowohl die Datenbank-Schnittstelle als auch den PaymentHandler-Client, um an die entsprechenden Daten aus der Datenbank bzw. vom PaymentHandler zu gelangen. Die Komponente DDS-Authoring benutzt die Datenbank-Schnittstelle, um die eingegebenen Daten in die Datenbank zu schreiben bzw. um bereits vorhandene Daten aus der Datenbank auszulesen.

In der ersten Programmierphase wurden die einzelnen Komponenten unabhängig voneinander ausprogrammiert, wobei die Schnittstellen-Methoden durch Dummy-Methoden zur Datenbeschaffung und Datenbereitstellung ersetzt wurden. Während der Integrationsphase wurden dann die Dummy-Methoden durch die Schnittstellen-Methoden ersetzt und somit die Integration der fünf Komponenten zum eigentlichen DDS-System nach und nach realisiert.

2.5 Installationsanleitung

Was braucht man, um das System einsetzen zu können?

Zum Betrieb des DDS-iShops ist neben der DDS-System-Software selbst der Apache-WWW-Server (mit Servlet-Unterstützung) und ein JDBC-unterstützendes Datenbanksystem erforderlich. Zusätzlich sollte genügend Speicherplatz zur Ablage der angebotenen hypermedialen Dokumente auf dem WWW-Server zur Verfügung stehen.
Für den Betrieb des DDS-Authoring-Tools sollte Java in der Version 1.2 oder höher installiert sein. Zusätzlich muss noch das Klassenarchiv der verwendeten Datenbank (hier: der Oracle-Datenbank) vorhanden sein, welches im CLASSPATH auftauchen muss.
Auf Seiten des Kunden sollte ein Standard-WWW-Browser mit eingeschaltetem JavaScript und dem Aktzeptieren von Cookies vorhanden sein. Für die Bezahlung mittels ECash oder CyberCash sollten entsprechende Wallets auf Kundenseite vorhanden sein.

Installationsanleitung:

Zunächst einmal muss auf dem Server des Anbieters ein Apache-WWW-Server mit Servlet-Unterstützung installiert werden, falls dieser noch nicht vorhanden ist. Die Dokumentenschutz-Komponente erweitert diesen mittels entsprechender Plug-ins bzw. eines C-Moduls und eines Servlets (Security.class) um die entsprechenden Dokumentenschutz-Mechanismen. Nachdem der Apache-WWW-Server auf dem Server des Anbieters installiert ist, muss zunächst das Server-Plugin in den Server-Kernel einkompiliert werden. Anschliessend sind noch einige Modifikationen zur Konfiguration des Web-Servers notwendig, die in den Dateien zone.properties und http.conf vorgenommen werden müssen. Das genaue Vorgehen bei der Installation der Dokumentenschutz-Komponente wird in **Kapitel 5: Dokumentenschutz-Komponente** beschrieben.

Die verwendeten Servlets der DDS-Web-Komponente und die verwendeten .class-Dateien wurden in ein JAR-File gepackt, welches auf dem Server abgelegt werden muss. Danach sind noch einige Anpassungen in den zone.properties und jserv.properties durchzuführen (insbesondere muss hier eingetragen werden, wo das JAR-File mit den Servlets abgelegt wurde), bevor der WebShop durch Aufruf der entsprechenden URL in einem Standard-WWW-Browser des Kunden genutzt werden kann. Nähere Informationen zur Installation der DDS-Web-Komponente finden Sie im **Kapitel 7: Web-Komponente**.

Bevor die Datenbankanbindung im DDS-System genutzt werden kann, müssen zunächst ein paar Anpassungen der Properties in der Datei Webshop.ini durchgeführt werden. Anschliessend müssen in der zugrundeliegenden Datenbank die benötigten Tabellen und Constraints zunächst mittels Installationsskript (dds-db.sql) angelegt werden, bevor die Datenbank über die JDBC-Schnittstelle durch das Authoring-Tool mit Daten gefüllt werden kann. Für weitere Informationen zu den Voraussetzungen und der Installation der Datenbank-Komponente lesen Sie bitte im **Kapitel 3: Datenbank-Komponente** nach.

Die relevanten Klassen der Payment-Komponente für den Server bzw. für den RMI-Client befinden sich in der Datei PHServer.jar. Für die Benutzung der verschiedensten Zahlungsverfahren sind zum Teil noch einige Anmelde- und Registrierungs-Formalitäten bei Banken, Kreditinstituten und Anbietern notwendig (siehe Kapitel: Payment-Komponente). Genaue Informationen zur Installation der DDS-Payment-Komponente finden Sie im **Kapitel 4: Payment-Komponente**.

Für das Authoring-Tool werden entsprechende Skript-Dateien für Unix und Windows zur Verfügung gestellt, die das Tool starten (atool.sh und atool.bat). Diese Skripte müssen noch ein wenig angepasst werden. In der Datei webshop.ini muss vom Benutzer die Anbindung des Authoring-Tools an die Datenbank, sowie Informationen zum verwendeten Payment-Handler angepasst werden. Wichtig ist noch, dass der Pfad zum JAR-Archiv und zur Klassenbibliothek der Datenbank im CLASSPATH auftauchen. (Nähere Informationen zur Installation des Authoring-Tools finden Sie unter dem **Kapitel 6: Authoring-Komponente**.)

Damit die angebotenen Dokumente auch über die Dokumentenschutz-Komponente vor dem nicht-autorisierten Zugriff geschützt sind, müssen diese Dokumente auf dem WWW-Server abgelegt werden und dürfen sich nicht irgendwo auf einem anderen Server im WWW befinden, wo sie durch die Dokumentenschutz-Komponente nicht geschützt werden können. Dies kann mittels eines normalen FTP-Programms geschehen.

Kapitel 3

Datenbank-Komponente

3.1 Datenbank-Entwurf

3.1.1 ER-Schemata

Die im Folgenden dargestellten Entity-Relationship-Modelle spiegeln die Relationen des Daten-Modells zu größten Teil direkt wider. Die Namen der Entitäten wurden für die entsprechenden Relationen bzw. Tabellen übernommen. Bezüglich des vollständigen DDS-Systems sind noch zahlreiche zusätzliche Entitäten denkbar. Da diese allerdings nicht persistent sein müssen, werden sie im Zusammenhang mit dem Datenbank-Entwurf nicht weiter betrachtet.

3.1.1.1 DDS-ER-Schema

Die in Abbildung 3.1 dargestellte Entität *Shop_Settings* beschreibt einen Tupel von DDS-Attributen, die in der Tabelle 3.14 näher erläutert werden. Zu diesen Einstellungen gehören ebenso eine Anzahl von unterstützten Zahlungsverfahren (siehe Entität *Brand*).

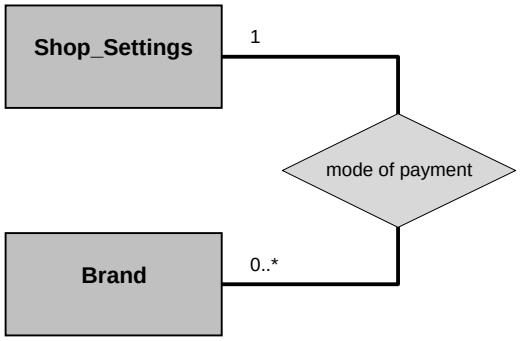

Abbildung 3.1: Entity-Relationship Diagramm für die DDS-Einstellungen

Das in Abbildung 3.2 dargestellte ER-Modell zeigt die Entitäten bzw. Objekte, welche den eigentlichen Inhalt des DDS bilden, sowie Objekte, die die Grundlage für die Ablauf- und

Zugriffssteuerung sowie die Verwaltung dieser Inhalte bilden.

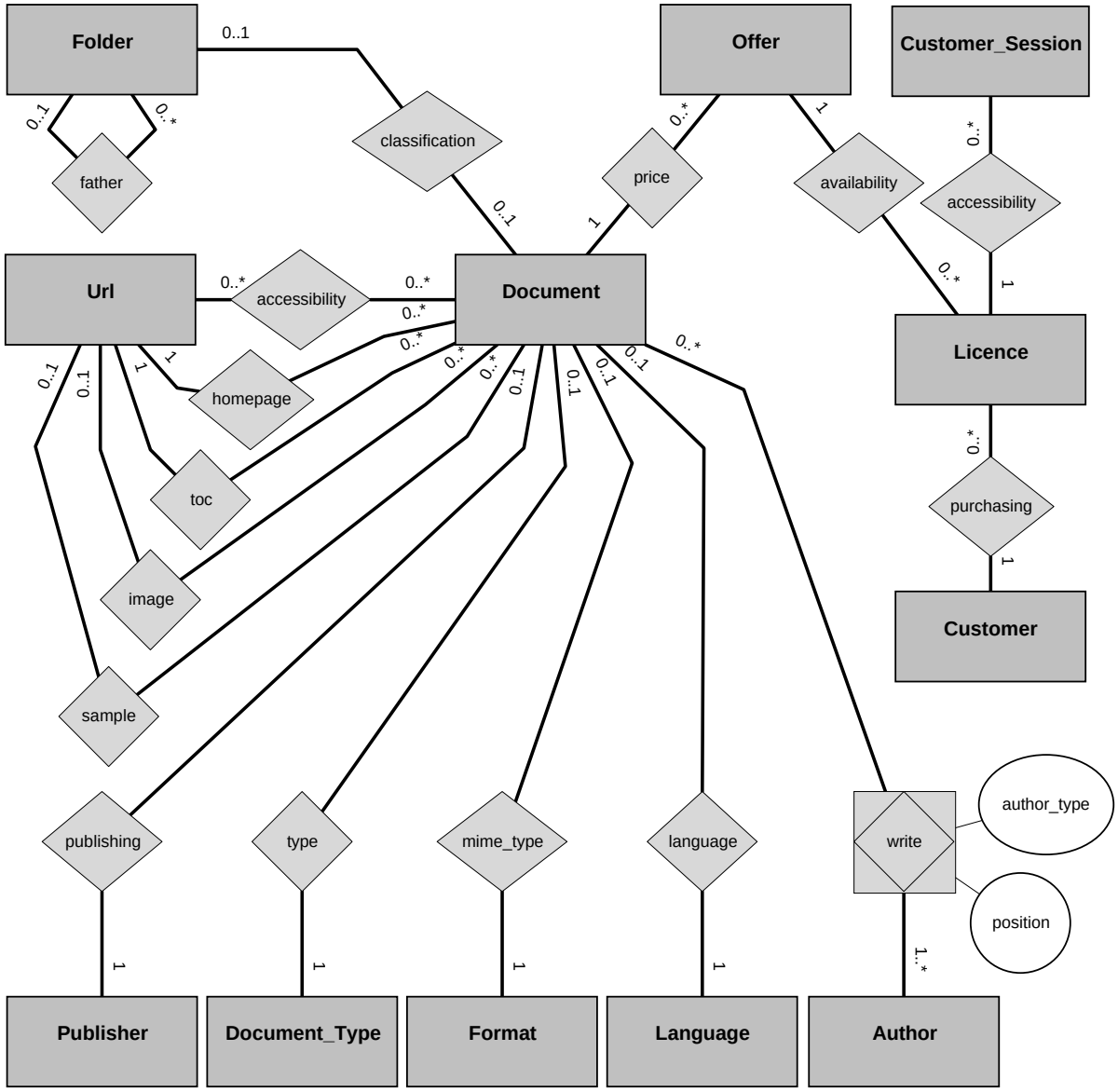

Abbildung 3.2: Entity-Relationship Diagramm für den DDS

Die „digitalen Produkte“ des DDS sind hypermediale Dokumente (*Document*; s. Tabelle 3.13), deren Zugriff über eine oder mehrere Urls erfolgen muß (*Url*; s. Tabelle 3.9). Für die Präsentation und Unterstützung des Kunden bei seiner Kaufentscheidung werden speziell ausgezeichnete Urls genutzt, die auf Dokumentinhalte unterschiedlichen Typs verweisen. Hierzu gehören die Anfangsseite (*homepage*), das Inhaltsverzeichnis (*toc*), eine Bilddatei (*image*) und eine Leseprobe (*sample*). Diese vier Elemente (ggf. auch noch weiterführende Links) sollen frei zugänglich sein. Alle weiteren Urls können geschützt werden, wenn der Anbieter dies verlangt. Sie verweisen

auf die eigentlichen kommerziellen Inhalte des Dokuments (Content). Die Entititäten Verfasser, Typ, Format, Sprache sowie Autor (*Publisher, Document_Type, Format, Language, Author*; s. Tabellen 3.3, 3.4, 3.5, 3.6 und 3.7) sind aus dem eigentlichen Dokument ausgelagert. Sie werden durch die DDS-Authoring Komponente separat zum Dokument (in eigenen „Pools“) verwaltet und können so gleichzeitig mehreren Dokumenten zugeordnet werden. Da ein Dokument u.U. mehrere Autoren besitzt, die evtl. auch in unterschiedlichen Funktionen beteiligt waren, werden sie in ihrer Dokument-Assoziation durch spezielle Attribute (*position, author_type*) unterschieden.

Für die Klassifikation der Dokumente sind so genannte Kataloge bzw. Ordner vorgesehen (*Folder*; s. Tabelle 3.1). Dabei kann ein Dokument auch verschiedenen Ordnern zugeordnet sein. Die Ordner selbst können wiederum hierarchisch organisiert werden. Der Zugriff auf das Angebot des Shops an digitalen Dokumenten erfolgt über die Ordner. Ein Dokument, das keinem der Kataloge zugeordnet ist, kann für diese Zeit auch nicht über den Shop erworben werden. Bestehende gültige Lizenzen für dieses Dokument laufen aber weiter.
Dem Kunden wird ein Dokument kostenpflichtig zugänglich gemacht. Der Preis wird in der Entität Angebot (*Offer*; s. Tabelle 3.2) festgehalten. Es ist möglich mehrere Angebote zu einem Dokument vorzuhalten, wovon aber immer nur eines aktuell gültig ist. Später erwirbt der Kunde dann Zeitlizenzen (*Licence*; s. Tabelle 3.10) für die angebotenen Dokumente. Er erkauft sich somit das Recht, ein Dokument solange abrufen zu können, bis die Zeit, für die er bezahlt hat, abgelaufen ist. Das Lesen von Dokumentinhalten erfolgt innerhalb einer oder mehrerer Kundensitzungen (*Customer_Session*; s. Tabelle 3.11), innerhalb derer die Zugriffssteuerung für eine Lizenz geregelt wird.

3.1.1.2 Payment-Handler-ER-Schema

Abbildung 3.3 zeigt das ER-Modell des zentralen Payment-Handlers. Als persistente Objekte (Entitäten) werden hier Transaktionen (*Transaction*; s. Tabelle 3.18), Web-Shops (*Shop*; s. Tabelle 3.16) und Kunden eines Web-Shops (*Payment_Customer*, s. Tabelle 3.17) verwaltet.

Anbieter bzw. Shopbetreiber, die den Payment-Handler nutzen, müssen registriert sein. Die Kunden eines Anbieters sind für den Payment-Handler anonym, was die Speicherung persönlicher Daten betrifft. Ein Kunde tritt dem Payment-Handler auch immer nur in Verbindung mit dem selben Shop gegenüber - ist er gleichzeitig auch noch Kunde eines anderen Anbieters, der ebenfalls die Dienste des Payment-Handlers in Anspruch nimmt, gilt er bei Geschäften, die er über diesen Anbieter tätigt, als neuer Kunde des Payment-Handlers. Als Kunde kann er über den Payment-Handler Transaktionen unterschiedlichen Typs abwickeln, deren Verlauf und Erfolg vom Payment-Handler protokolliert werden und die für den Kunden jederzeit über den Umweg des (DDS-) Shops einsehbar sein sollen.

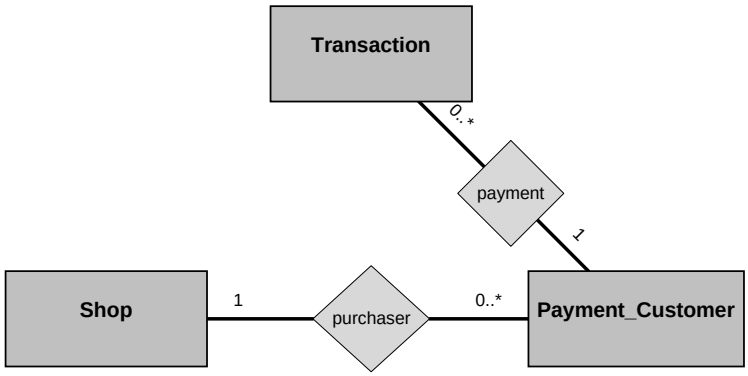

Abbildung 3.3: Entity-Relationship Diagramm für den Payment-Manager

3.1.2 DB-Tabellendefinitionen

3.1.2.1 DDS-Tabellen

Folder: Die Tabelle Folder enthält die Kategorien für die Klassifizierung der Digitalen Dokumente. Die Kategorien selbst sind in einer Baumstruktur organisiert, was eine feinere Untergliederung ermöglicht.

Attributname	Domain	Beschreibung
folder_id	NUMBER	ID des Folders (PK)
name	STRING	Name des Folders bzw. der Kategorie
father_folder_id	NUMBER	folder_id des übergeordneten Folders

Tabelle 3.1: Tabelle Folder

Offer: Die Tabelle Offer enthält Angebote zu den digitalen Dokumenten. Zu jedem Dokument können unterschiedliche Angebote existieren. Aus Käufersicht ist aber immer nur ein Angebot aktuell gültig. Angebote können im nachhinein nicht mehr verändert (ausgenommen die Gültigkeit) oder gelöscht werden.
Die Angebotsform ist bewusst sehr restriktiv gehalten. Es ist z.B. nicht möglich eine Benuzungsdauer für einen Angebotspreis festzulegen. Die Benuzungsdauer soll für alle Angebote gleich sein und wird daher in der Implementierung der DDS-Applikationen festgelegt.

Attributname	Domain	Beschreibung
offer_id	NUMBER	ID des Angebots (PK)
doc_id	NUMBER	ID des Dokuments, für das dieses Angebot gilt (FK)
price	NUMBER	Angebotspreis
currency	STRING	Währung des Angebotspreises
valid_offer	NUMBER	Gültigkeit des Angebots zum aktuellen Zeitpunkt
settled_date	NUMBER	Datum/Zeit, an dem das Angebot angelegt wurde

Tabelle 3.2: Tabelle Offer

Publisher: Die Tabelle Publisher speichert die Verleger der diditalen Dokumente. Sie bildet einen Pool von Verlagen, die den Dokumenten im DDS zugeordnet werden können, was ein einheitliches Erscheinungsbild der Verlagsbezeichnungen über alle Dokumente hinweg begünstigen soll. Das Attribut ist dem *Dublin Core Element Set* entnommen.

Attributname	Domain	Beschreibung
publisher_id	NUMBER	ID des Verlags (PK)
publisher_name	STRING	Bezeichnung des Verlags

Tabelle 3.3: Tabelle Publisher

Document_Type: Die Tabelle Document_Type speichert die unterschiedlichen Dokumenttypen. Sie bildet einen Pool Dokumenttypen, was eine einheitliche Schreibweise bei allen Dokumenten unterstützen soll. Das Attribut ist dem *Dublin Core Element Set* entnommen.

Attributname	Domain	Beschreibung
type_id	NUMBER	ID des Dokumenttyps (PK)
doc_type	STRING	Bezeichnung des Typs

Tabelle 3.4: Tabelle Document_Type

Format: Die Tabelle Format speichert die unterschiedlichen Dokumentformate. Da es sich ausschließlich um digitale Dokumente handelt, bildet sie einen Pool von Mime-Types, die den Dokumenten im DDS zugeordnet werden können, was ein einheitliches Erscheinungsbild und eine korrekte Zuordnung des jeweiligen Formats für alle Dokumente begünstigen soll. Das Attribut ist dem *Dublin Core Element Set* entnommen.

Attributname	Domain	Beschreibung
format_id	NUMBER	ID des Dokumentformats (PK)
doc_format	STRING	Bezeichnung des Formats

Tabelle 3.5: Tabelle Format

Language: Die Tabelle Language speichert Sprachen. Sie bildet einen Pool von Sprachen, die den Dokumenten im DDS zugeordnet werden können, was eine einheitliche Schreibweise für alle Dokumente begünstigen soll. Das Attribut ist dem *Dublin Core Element Set* entnommen.

Attributname	Domain	Beschreibung
language_id	NUMBER	ID der Sprache (PK)
language	STRING	Sprache, in der ein Dokument verfasst sein kann

Tabelle 3.6: Tabelle Language

Author: Die Tabelle Author enthält den Autorenstamm der Dokumente im DDS. Die Autoren können verschiedenen Dokumenten zugeordnet sein, was unterschiedliche Schreibweisen verhindert und einer bessere Trefferquote bei einer Suche über Autoren begünstigt.

Attributname	Domain	Beschreibung
author_id	NUMBER	ID des Autors (PK)
name	STRING	Bezeichnung bzw. der Name des Autors

Tabelle 3.7: Tabelle Author

Document_Author: Die Tabelle Document_Author dient der Assoziation zwischen Dokumenten und Ihren Autoren. Als zusätzliche Attribute der Assoziation werden die Rolle des Autors sowie seine Position in einer Reihe mehrerer Autoren festgehalten.

Attributname	Domain	Beschreibung
doc_id	NUMBER	ID des Dokuments, für das dieses Angebot gilt (FK)
author_id	NUMBER	ID eines Autors des Dokuments(FK)
author_type	STRING	Bezeichnung der Rolle des Autors
position	NUMBER	Position in der Reihe mehrerer Autoren eines Dokuments

Tabelle 3.8: Tabelle Document_Author

Url: Die Tabelle Url enthält die relativen URLs der digitalen Dokumente auf dem Server. URLs, bei denen das *free*-Attribut gesetzt ist, umgehen das Dokumenten-Sicherheitsmanagement und sind frei zugänglich.

Attributname	Domain	Beschreibung
url_id	NUMBER	ID des Autors (PK)
address	STRING	relativer Teil einer Url des Dokuments
free	NUMBER	Kennzeichnung einer kostenfreien Url

Tabelle 3.9: Tabelle Url

Licence: Die Tabelle Licence enthält die von den Kunden erworbenen Lizenzen zu den jeweiligen Dokumenten. Lizenzen beziehen sich dabei nicht direkt auf ein Dokument, sondern immer auf ein mit dem Dokument verknüpftes Angebot. Lizenzen sind zeitbezogen, d.h. nur für eine bestimmte Dauer gültig. Greift ein Kunde das erste Mal auf seine Lizenz zu, wird dieser Zeitpunkt bei *first_click* eingetragen und die Lizenzdauer beginnt abzulaufen. Der Zugriff darf erst dann erfolgen, wenn das *valid*-Kennzeichen für den Status einer Bezahlung gesetzt wurde. Um diesen Status auszulesen, muss der Payment-Manager mit der *phPay_id* kontaktiert werden.

Attributname	Domain	Beschreibung
licence_id	NUMBER	ID der Lizenz (PK)
customer_id	NUMBER	ID des Kunden, der die Lizenz erwarb (FK)
offer_id	NUMBER	ID des Angebots, zu dem diese Lizenz erworben wurde (FK)
phPay_id	NUMBER	ID der Transaktion, mit der die Lizenz bezahlt wurde. (Die ID wird extern, vom Payment-Manager vergeben.)
duration	NUMBER	Dauer der Lizenz in einer Zeiteinheit
first_click	NUMBER	Datum/Zeit des ersten Zugriffs auf die Lizenz
valid	NUMBER	Kennzeichnung, dass die Lizenz bereits bezahlt wurde

Tabelle 3.10: Tabelle Licence

Customer_Session: Die Tabelle Customer_Session enthält Daten für den Dokumenten-Zugriffsschutz. Greift ein Kunde auf ein Dokument zu und besitzt eine gültige Lizenz, wird von einem Sicherheits-Manager eine Sitzung erzeugt und die Sitzungsdaten in diese Tabelle eingetragen. Die Sitzung stellt ein einfacheres Zugiffs- und Sicherheitskonzept dar, denn solange eine Sitzung gültig ist (die Dauer wird vom Sicherheits-Manager vorgegeben), ist eine Authentifizierung des Kunden und eine Überprüfung der Lizenz unnötig. Hier soll die *session_id* als Zugriffsschlüssel für die Lizenz dienen.

Attributname	Domain	Beschreibung
session_id	NUMBER	ID der Sitzung (PK)
licence_id	NUMBER	ID der Lizenz, die in der Sitzung abgerufen wird (FK)
valid_to	NUMBER	Datum/Zeit bis die Sitzung ungültig wird

Tabelle 3.11: Tabelle Customer_Session

Customer: Die Tabelle Customer enthält die Kundendaten des DDS. Erforderliche Daten eines Kunden sind hier aber lediglich *login* und *password*. Alle anderen Angeben sind freiwillig und dienen z.T. nur statistischen Zwecken. Jeder Nutzer, der ein Lizenz für ein Dokument erwerben will, wird zuvor in diese Tabelle aufgenommen. Für die Bezahlungsabwicklung muss die *customer_id* des Kunden auch an den Payment-Manager weitergegeben werden.

Attributname	Domain	Beschreibung
customer_id	NUMBER	ID des Kunden (PK)
name	STRING	Nachname
first_name	STRING	Vorname(n)
address	STRING	Straße mit Hausnummer
city	STRING	Wohnort
p_code	STRING	Postleitzahl
country	STRING	Land
email	STRING	e-Mail-Adresse
phone	STRING	Telefonnummer
fax	STRING	Telefaxnummer
login	STRING	Benutzername innerhalb des Systems
password	STRING	Passwort zur Anmeldung am System

Tabelle 3.12: Tabelle Customer

Document: Die Tabelle Document enthält Daten über die im DDS angebotenen digitalen Dokumente. Neben Elementen des *Dublin Core Element Set* finden sich hier auch Referenzen auf frei zugängliche URLs, die dem Kunden zusätzliche inhaltliche Informationen in Form von Leseproben etc. zur Verfügung stellen sollen.

Attributname	Domain	Beschreibung
doc_id	NUMBER	ID des Dokuments (PK)
title	STRING	Dokumenttitel
description	STRING	Kurzbeschreibung des Inhalts
publisher_id	NUMBER	ID des Verlags (FK)
pyear	NUMBER	Veröffentlichungsjahr
type_id	NUMBER	ID des Dokumenttyps (FK)
format_id	NUMBER	ID des Dokumentformats (FK)
language_id	NUMBER	ID der Dokumentsprache (FK)
isbn	STRING	Internationale Standard Buchnummer
edition	NUMBER	Nummer der Auflage
homepage	NUMBER	ID der Url der Startseite des Dokuments (FK)
toc	NUMBER	ID der Url des Inhaltsverzeichnisses des Dokuments (FK)
image	NUMBER	ID der Url eines Icons, Logos oder Erkennungsbildes des Dokuments (FK)
sample_homepage	NUMBER	ID der Url der Startseite einer Leseprobe (FK)

Tabelle 3.13: Tabelle Document

Shop_Settings: Die Tabelle Shop_Settings speichert die allgemeinen Shopeinstellungen des DDS sowie Informationen des Shopbetreibers. Diese Tabelle wird nur sehr statisch benutzt, d.h. sie enthält nur einen einzigen Datensatz für die Shopeinstellungen, da die gesamte Datenbank auch nur einen Shop repräsentiert.

Attributname	Domain	Beschreibung
shop_id	NUMBER	ID der Shopeinstellungen (PK)
shop_name	STRING	offizieller Name des Shops
email	STRING	e-mail-Adresse der Shopbetreiber (Kontakt)
layout	NUMBER	Nummer eines vorgefertigten Layouts für das WWW
owner_name	STRING	Name des Shopbetreibers
address	STRING	Adresse des Shopbetreibers
phone	STRING	Telefonnummer des Shopbetreibers
fax	STRING	Telefaxnummer des Shopbetreibers
agb_txt_URL	STRING	Url bzw. vollständiger Pfad einer Textdatei mit den allgemeinen Geschäftsbedingungen des Shopbetreibers
shop_info_URL	STRING	Url einer Info-Seite zum Shop
start_page_URL	STRING	Url des Shops im WWW
logo_URL	STRING	Url zu einer Bilddatei des Shop-Logos
root_path	STRING	absoluter Pfad zu den Dokumenten auf dem Server

Tabelle 3.14: Tabelle Shop_Settings

Brand: Die Tabelle Brand enthält die vom DDS unterstützten Zahlungssysteme. Es werden nur die vom Payment-Handler vergebenen IDs verwaltet und mit den Shopeinstellungen verknüpft. Inhaltliche Informationen zu den Zahlungssystemen werden nicht gespeichert, da sie prinzipiell vom verwendeten Payment-Handler abhängig sind.

Attributname	Domain	Beschreibung
brand_id	NUMBER	ID des Zahlungssystems (PK) (Die ID wird extern vom Payment-Manager vergeben.)
shop_id	NUMBER	ID der Shopeinstellungen (FK)

Tabelle 3.15: Tabelle Brand

3.1.2.2 Payment-Handler-Tabellen

Shop: Die Tabelle Shop nimmt die erforderlichen Daten eines Shopbetreibers auf, der die Dienste des Payment-Handlers nutzen möchte. Bei Angabe einer Bankverbindung kann der Payment-Handlers dem Shopbetreiber die jeweiligen Einnahmen direkt auf seinem Konto gutschreiben.

Attributname	Domain	Beschreibung
shop_id	NUMBER	ID des Shops (PK)
name	STRING	offizieller Name des Shops
password	STRING	Zugangspasswort des Shops
bank_connection	STRING	Bankverbindung

Tabelle 3.16: Tabelle Shop

Payment_Customer: Die Tabelle Payment_Customer nimmt die erforderlichen Kundendaten eines Shopbetreibers auf, wenn er die Dienste des Payment-Handlers nutzt. Außerdem werden hier die Salden der persönlichen Kundenkonten gespeichert. Für den Payment-Handler selbst bleibt der Kunde relativ anonym, abgesehen von persönlichen Daten, die beispielsweise für die Durchführung eines Lastschriftverfahrens anzugeben sind, aber nicht persistent gespeichert werden. Ein realer Kunde sollte auch niemals mit dem Payment-Handler selbständig in Kontakt treten. Statt dessen muss seine Funktionalität durch die Shopbetreiber gekapselt und dem Kunden über einfache Schnittstellen zu Verfügung gestellt werden.

Attributname	Domain	Beschreibung
shop_customer_id	NUMBER	ID des Kunden (PK)
shop_id	NUMBER	ID des Shops (FK)
balance	NUMBER	Saldo des Kundenkontos
credit	NUMBER	eingeräumter Überziehungskredit
customer_id	NUMBER	ID des Kunden innerhalb eines Shops

Tabelle 3.17: Tabelle Payment_Customer

Transaction: Die Tabelle Transaction speichert sämtliche Transaktionen, die vom Payment-Handler bearbeitet werden. Jede Transaktion wird von einem *Payment_Customer* ausgelöst. Eine Transaktion durchläuft, bis sie als beendet gilt, verschiedene Zustände, die durch die Belegung Attribute in der Tabelle widergespiegelt werden. Diese Attribute sollen vom Payment-Handler zeitnah aktualisiert werden.

Attributname	Domain	Beschreibung
phPay_id_id	NUMBER	ID der Transaktion (PK)
taType	NUMBER	
taStatus	NUMBER	Status der Transaktion
errorStatus	NUMBER	Fehlercode
cOrg_id	NUMBER	Identifikator des Kunden, z.B. IP-Adresse, Login
shortDesc	STRING	Kurzbeschreibung der Transaktion bzw. Verwendungszweck
invoicedAt	NUMBER	Zeitpunkt der Zahlungsaufforderung
invoicedAmount	NUMBER	Betrag der Zahlungsaufforderung
paidAt	NUMBER	Zeitpunkt der Zahlung
paidAmount	NUMBER	Betrag der Zahlung
settledAt	NUMBER	Zeitpunkt der Buchung

Tabelle 3.18: Tabelle Transaction

3.2 Datenbank-Schnittstellen

3.2.1 Die Schnittstellen von DB_Access

Die Klassen, die die Schnittstellen der Datenbank-Komponente implementieren, sind im Package *webshop.dbif* zusammengefasst. Dazu gehört als Schnittstelle für alle Konmponenten des DDS-Systems die Klasse *webshop.dbif.DB_Access*. Sie implementiert die Methoden zum Auslesen von Datensätzen und zur Durchführung von Transaktionen auf der Datenbank. Der Payment-Handler erhält über eigene Methoden Zugriff auf die Datenbank. Die Komponenten des DDS-Sytems verwenden teilweise dieselben Methoden. Verschiedene Ausnahmen- und Fehlerklassen sind für die Fehlerbehandlung zuständig, die Verbindungen zu der Datenbank organisiert die Klasse *webshop.dbif.DBMediatior*.

3.2.2 Die Schnittstellen für das DDS-System

Die Klasse *webshop.dbif.DB_Access* stellt eine Reihe von Methoden zur Verfügung, mit denen Datensätze aus der Datenbank ausgelesen, eingefügt sowie geändert werden können. Es handelt sich bei allen Methoden um Klassenmethoden („static“-Methoden).

getFolders Holt aus der Tabelle Folder alle Folderobjekte, die keinen übergeordneten Folder haben.

```
public static java.util.Vector getFolders()
throws InternalDbifError
```

Returns: ein Vektor mit Objekten vom webshop.classes.Folder
Throws: InternalDbifError - Schwerer Fehler in webshop.dbif

getSubFolders Holt aus de Tabelle Folder alle untergeordneten Folderobjekte zu dem Folder mit der ID fatherId.

```
public static java.util.Vectorn getSubFolders(int fatherId)
throws InternalDbifError, InvalidIDException
```

Parameters: fatherId - die ID des Folders, zu dem die Subfolder zurueckgegeben werden sollen
Returns: ein Vektor mit Objekten vom webshop.classes.Folder
Throws: InternalDbifError - Schwerer Fehler in webshop.dbif
InvalidIDException - Es existiert kein Folder zu dem Parameter fatherId

getFolder Holt den Folder mit PK gleich dem Parameter folderId aus der Tabelle Folder.

```
public static Folder getFolder(int folderId)
throws InternalDbifError, InvalidIDException
```

Parameters: folderId - PK in der Tabelle Folder
Returns: webshop.classes.Folder
Throws: InternalDbifError - Schwerer Fehler in webshop.dbif
InvalidIDException - Es existiert kein Folder zu dem Parameter folderId

getDocumentInfo Holt ein DocumentInfo Objekt zum Dokument mit der ID im Parameter intdocId.

```
public static DocumentInfo getDocumentInfo(intdocId)
throws InternalDbifError, InvalidIDException
```

Parameters: intdocId - PK in der Tabelle Document
Returns: webshop.classes.DocumentInfo
Throws: InternalDbifError - Schwerer Fehler in webshop.dbif
InvalidIDException - Es existiert kein Folder zu dem Parameter intDocId

getDocumentInfos Holt einen Vector mit den DocumentInfos aller Dokumente in dem Folder mit der ID des Parameters folderId. Wenn der Folder leer ist, wird ein leerer Vektor zurückgegeben.

```
public static java.util.Vector getDocumentInfos(int folderId)
throws InternalDbifError, InvalidIDException
```

Parameters: folderId - PK in der Tabelle Folder
Returns: Vektor mit Objekten vom Typ webshop.classes.DocumentInfo
Throws: InternalDbifError - Schwerer Fehler in webshop.dbif
InvalidIDException - Es existiert kein Folder zu dem Parameter folderId

getDocument Holt ein Document Objekt anhand der docId.

```
public static Document getDocument(int docId)
throws InternalDbifError, InvalidIDException
```

Parameters: docId - die Document ID
Returns: webshop.classes.Document
Throws: InternalDbifError - Schwerer Fehler in webshop.dbif
InvalidIDException - Es existiert kein Document zum Parameter docId.

getCustomer Holt ein Costomer Objekt anhand der costumerId.

```
public static Customer getCustomer(int customerId)
throws InternalDbifError, InvalidIDException
```

Parameters: customerId - die ID des Customers
Returns: webshop.classes.Customer
Throws: InternalDbifError - Schwerer Fehler in webshop.dbif
InvalidIDException - Es existiert kein Customer zum Parameter customerId.

getCustomer Holt ein Costomer Objekt anhand des Logins und des Passwortes.

```
public static Customer getCustomer(java.lang.String login,
                                   java.lang.String password)
throws InternalDbifError, InvalidLoginException
```

Parameters: login - das Login des Customers, password - das Passwort des Customers
Returns: webshop.classes.Customer
Throws: InternalDbifError - Schwerer Fehler in webshop.dbif
InvalidLoginException - Es existiert kein Customer zur login-password Kombination

updateCustomer Aktualisiert die Daten zu einem Kunden.

```
public static void updateCustomer(Customer customer)
throws InternalDbifError, InvalidIDException
```

Parameters: customer - webshop.classes.Customer
Throws: InternalDbifError - Schwerer Fehler in webshop.dbif
InvalidIDException - Die ID des Customers ist ungültig

insertCustomer Fügt einen neuen Customer in die Datenbank ein.

```
public static int insertCustomer(Customer customer)
throws InternalDbifError
```

Parameters: customer - webshop.classes.Customer
Returns: Die zugewiesene customer_id (PK)
Throws: InternalDbifError - Schwerer Fehler in webshop.dbif

getValidLicences Holt alle noch gültigen Lizenzen eines Kunden mit der ID customerId. Wenn der Kunde keine gültigen Lizenzen mehr besitzt, wird ein leerer Vektor zurückgegeben. Gültige Lizenzen sind dabei solche, deren Dauer seit dem ersten Klick bis zur aktuellen Systemzeit noch nicht abgelaufen ist.

```
public static java.util.Vector getValidLicences(int customerId)
throws InternalDbifError, InvalidIDException
```

Parameters: customerId - PK in der Tabelle Customer
Returns: Vector mit Objekten vom Typ webshop.classes.LicenceInfo
Throws: InternalDbifError - Schwerer Fehler in webshop.dbif
InvalidIDException - Es existiert kein Customer zum Parameter customerId

getAllLicences Holt alle Lizenzen eines Kunden mit der ID customerId. Wenn der Kunde keine Lizenzen besitzt, wird ein leerer Vektor zurückgegeben. Bei abgelaufenen Lizenzen ist der Wert für die Restzeit (duration) negativ (seit X ms abgelaufen).

```
public static java.util.Vector getAllLicences(int customerId)
throws InternalDbifError, InvalidIDException
```

Parameters: customerId - PK in der Tabelle Customer
Returns: Vector mit Objekten vom Typ webshop.classes.LicenceInfo
Throws: InternalDbifError - Schwerer Fehler in webshop.dbif
InvalidIDException - Es existiert kein Customer zum Parameter customerId

doSearch Holt alle DocumentInfos zu dem Suchkriterium.

```
public static java.util.Vector doSearch(SearchCriteria searchCriteria)
throws InternalDbifError
```

Parameters: searchCriteria - ein Suchkriterium
Returns: Vector mit Objekten vom Typ webshop.classes.DocumentInfo
Throws: InternalDbifError - Schwerer Fehler in webshop.dbif

insertLicence Fügt eine neue Kunden-Lizenz für ein Angebot in die Datenbank ein. Der first_Click sowie das valid-flag werden dabei auf 0 gesetzt. Gleichzeitig wird zur neuen Lizenz eine Session_Id in die Tabelle Customer_Session geschrieben. Valid_to wird für diese Session auf 0 gesetzt.

```
public static int insertLicence(int phPayId, int customerId,
                                int offerId, long duration,
                                java.lang.String sid)
throws InternalDbifError, InvalidIDException, InvalidWriteIDException
```

Parameters: phPayId - die ID der Transaktion, mit der die Lizenz bezahlt wurde,
customerId - die ID des Customers,
offerId - die ID des Angebots,
duration - die Dauer der Lizenz,
sid - die erste Session_Id
Returns: Die zugewiesene ID für die neue Lizenz
Throws: InternalDbifError - Schwerer Fehler in webshop.dbif
InvalidWriteIDException - Die SessionID wurde schon für eine andere Session vergeben. (Kein eindeutiger Primärschluessel in der Tabelle Customer_Session)
InvalidIDException - Ungültige Fremdschlüsselbeziehung in einem oder mehreren der Parameter

setLicenceValid Kennzeichnet eine Lizenz als gültig, d.h. die Bezahlung ist erfolgreich abgeschlossen.

```
public static void setLicenceValid(int licenceId)
throws InvalidIDException, InternalDbifError
```

Parameters: licenceId - Die ID (PK) der Lizenz
Throws: InvalidIDException - Es existiert keine Lizenz zum Parameter licenceId
InternalDbifError - Schwerer Fehler in webshop.dbif

getShopSettings Holt die Shop-Einstellungen für den Webshop. Standardmäßig werden die Einstellungen mit der in webshop.dbif.DB_Access.SHOP_ID festgelegten ID aus der Tabelle Shop_Settings geholt. Wenn noch keine BrandIds vergeben sind, ist der Vektor für die BrandIds in den ShopSettings leer.

```
public static ShopSettings getShopSettings()
throws InternalDbifError
```

Returns: webshop.classes.ShopSettings
Throws: InternalDbifError - Schwerer Fehler in webshop.dbif

getLicenceSessionInfo Holt ein LicenceSessionInfo Objekt zu einer Url mit SessionId.

```
public static LicenceSessionInfo
                getLicenceSessionInfo(java.lang.String url,
                                      java.lang.String sessionId)
throws InternalDbifError, InvalidIDException, BadFitException
```

Parameters: url - address in der Tabelle Url,
sessionId - PK in der Tabelle Session
Returns: webshop.server.LicenceSessionInfo
Throws: InternalDbifError - schwerer Fehler in webshop.dbif
InvalidIDException - Es existiert keine Customer_Session zum Parameter sessionId.
BadFitException - Die SessionID gehoert nicht zur angegebenen Url

checkPassword Autentifiziert einen Kunden anhand seiner ID, seines Logins und seines Passworts.

```
public static boolean checkPassword(java.lang.String login,
                                    java.lang.String password,
                                    int customerId)
throws InternalDbifError, InvalidIDException
```

Parameters: login - Das Login des Kunden aus der Tabelle Customer
password - Das Passwort des Kunden aus der Tabelle Customer
customer_id - PK in der Tabelle Customer
Returns: TRUE, wenn alle Parameter mit einem Eintrag in der Tabelle Customer übereinstimmen, sonst FALSE
Throws: InternalDbifError - Schwerer Fehler in webshop.dbif
InvalidIDException - Es existiert kein Customer zu den Parametern.

getTaStatus Prüft, ob eine Transaktion erfolgreich abgeschlossen wurde.

```
public static boolean getTaStatus(int phPayId)
throws InternalDbifError, InvalidIDException
```

Parameters: phPay_id - PK in der Tabelle Transaction
Returns: TRUE, wenn die Transaction zu dem Parameter phPayId als erfolgreich gekennzeichnet ist, sonst FALSE
Throws: InternalDbifError - Schwerer Fehler in webshop.dbif
InvalidIDException - Es existiert keine Transaction zum Parameter phPayId.

setFirstClick Setzt den First_Click der Lizenz mit der im Parameter übergebenen ID licenceId auf die aktuelle Systemzeit, sofern First_Click nicht bereits vorher gesetzt wurde.

```
public static void setFirstClick(int licenceId)
throws InternalDbifError, InvalidIDException,
       InvalidSemanticException
```

Parameters: licenceId - PK in der Tabelle Licence
Throws: InternalDbifError - Schwerer Fehler in webshop.dbif
InvalidIDException - Es existiert keine Lizenz zum Parameter licenceId.
InvalidSemanticException - Das Update ist Fehlgeschlagen, weil First_Click bereits gesetzt war.

setSessionID Trägt eine neue Session in Tabelle Customer_Session ein. Dies darf erst dann geschehen, wenn die Lizenz durch Setzen des Fist_Click aktiviert wurde und nicht abgelaufen ist.

```
public static void setSessionID(java.lang.String sessionId,
                                int licenceId,
                                java.util.Date validTo)
throws InternalDbifError, InvalidIDException,
       InvalidWriteIDException
```

Parameters: sessionId - Die ID der Session in der Tabelle Customer_Session—
licenceId - Die ID der Lizenz aus der Tabelle Licence
validTo - Das Datum, an dem die Session abläuft
Throws: InternalDbifError - Schwerer Fehler in webshop.dbif InvalidIDException - Es existiert keine Lizenz zum Parameter licenseId
InvalidWriteIDException - Die SessionID wurde schon für eine andere Session vergeben. (Kein eindeutiger Primärschlüssel in der Tabelle Customer_Session)

getDocumentDetails Liefert die Details (Dublin Core) zu einem Dokument. Die ShopEntity Objekte Publisher, DocumentFormat, DocumentType und Language im DetailHolder sollten vor ihrer Verwendung auf NULL überprüft werden.

```
public static DetailHolder getDocumentDetails(int docId)
throws InternalDbifError, InvalidIDException
```

Parameters: docId - die ID (PK) des Dokuments
Returns: webshop.classes.DetailHolder
Throws: InternalDbifError - Schwerer Fehler in webshop.dbif
InvalidIDException - Es existiert kein Dokument zum Parameter docId

getDocumentUrls Liefert alle URLs zu einem Document. Wenn zu dem Dokument noch keine URLs in die Datenbank eingetragen wurden, wird ein leerer UrlHolder zurückgegeben.

```
public static UrlHolder getDocumentUrls(int docId)
throws InternalDbifError, InvalidIDException
```

Parameters: docId - die ID (PK) des Dokuments
Returns: webshop.classes.UrlHolder
Throws: InternalDbifError - Schwerer Fehler in webshop.dbif
InvalidIDException - Es existiert kein Dokument zum Parameter docId

getDocumentAuthors Liefert die Autoren zu einem Dokument mit der ID im Parameters docId.

```
public static AuthorHolder getDocumentAuthors(int docId)
throws InternalDbifError, InvalidIDException
```

Parameters: docId - die ID (PK) des Dokuments
Returns: webshop.classes.AuthorHolder
Throws: InternalDbifError - Schwerer Fehler in webshop.dbif
InvalidIDException - Es existiert kein Dokument zum Parameter docId

getDocumentOffers Liefert die Angebote zu einem Document.

```
public static OfferHolder getDocumentOffers(int docId)
throws InternalDbifError, InvalidIDException
```

Parameters: docId - die ID (PK) des Dokuments
Returns: webshop.classes.OfferHolder
Throws: InternalDbifError - Schwerer Fehler in webshop.dbif
InvalidIDException - Es existieren keine Angebote zum Dokument mit der ID docId oder das Dokument existiert nicht

getAllAuthors Liefert alle Autoren in der Datenbank. Wenn, die Tabelle keine Einträge enthält, ist der Rückgabewert ein leerer Vektor

```
public static java.util.Vector getAllAuthors()
throws InternalDbifError
```

Returns: Vektor mit Objekten vom Typ webshop.classes.Author
Throws: InternalDbifError - Schwerer Fehler in webshop.dbif

getAllPublishers Liefert alle Verlage in der Datenbank. Wenn, die Tabelle keine Einträge enthält, ist der Rückgabewert ein leerer Vektor.

```
public static java.util.Vector getAllPublishers()
throws InternalDbifError
```

Returns: Vektor mit Objekten vom Typ webshop.classes.Publisher
Throws: InternalDbifError - Schwerer Fehler in webshop.dbif

getAllDocumentTypes Liefert alle Document-Typen in der Datenbank. Wenn die Tabelle keine Einträge enthält, ist der Rückgabewert ein leerer Vektor.

```
public static java.util.Vector getAllDocumentTypes()
throws InternalDbifError
```

Returns: Vektor mit Objekten vom Typ webshop.classes.DocumentType
Throws: InternalDbifError - Schwerer Fehler in webshop.dbif

getAllLanguages Liefert alle Sprachen in der Datenbank. Wenn, die Tabelle keine Einträge enthält, ist der Rückgabewert ein leerer Vektor.

```
public static java.util.Vector getAllLanguages()
throws InternalDbifError
```

Returns: Vektor mit Objekten vom Typ webshop.classes.Language
Throws: InternalDbifError - Schwerer Fehler in webshop.dbif

getAllDocumentFormats Liefert alle Document-Formate in der Datenbank. Wenn die Tabelle keine Einträge enthält, ist der Rückgabewert ein leerer Vektor.

```
public static java.util.Vector getAllDocumentFormats()
throws InternalDbifError
```

Returns: Vektor mit Objekten vom Typ webshop.classes.DocumentFormat
Throws: InternalDbifError - Schwerer Fehler in webshop.dbif

insertDocument Fügt ein neues Dokument in die Tabelle Dokument ein und speichert zusätzlich alle Detailinformationen in den verknüpften Tabellen Offer, Document_Author sowie Document_Offer. Es werden nur vollständige Dokumente gespeichert.

```
public static int insertDocument(Document document)
throws InternalDbifError, IncompleteDataException
```

Parameters: document - webshop.Classes.Document kapselt die Informationen zu einem Dokument
Returns: Die zugewiesene ID (PK) für das neue Dokument
Throws: InternalDbifError - Schwerer Fehler in webshop.dbif
IncompleteDataException - Das neue Dokument enthält unvollständige Daten

updateDocumentDetails Aktualisiert die „DublinCore“ Elemente (details) zu einem Dokument mit der Id docID.

```
public static void updateDocumentDetails(int docId,
                                         DetailHolder details)
throws InternalDbifError, InvalidIDException,
       IncompleteDataException
```

Parameters: docId - Die ID (PK) des Dokuments, das aktualisiert werden soll
details - webshop.classes.DetailHolder
Throws: InternalDbifError - Schwerer Fehler in webshop.dbif

InvalidIDException - Es existiert kein Dokument zum Parameter docId
IncompleteDataException - Der DetailHolder enthält unvollständige Daten

updateDocumentUrls Aktualisiert die URLs zu einem Dokument mit der Id docID, zusätzliche Urls (mit Url_Id = 0) werden in die Tabellen Url bzw. Document_Url eingefügt und Urls, deren Löschkennzeichen gesetzt sind, werden aus der Tabelle Document_Url entfernt. Zusätzlich wird für jede Url, deren Änderungskennzeichen gesetzt ist, der free-Satus in der Datenbank aktualisiert. Die Änderungen sind für alle Dokumente übergreifend, die diese Url referenzieren.

```
public static void updateDocumentUrls(int docId, UrlHolder urls)
throws InternalDbifError, InvalidIDException, IncompleteDataException
```

Parameters: docId - Die ID (PK) des Dokuments, dessen Urls aktualisiert werden sollen
urls - webshop.classes.UrlHolder
Throws: InternalDbifError - Schwerer Fehler in webshop.dbif
InvalidIDException - Es existiert kein Dokument zum Parameter docId
IncompleteDataException - Der UrlHolder enthält unvollständige Daten

updateDocumentAuthors Aktualisiert die Autoren zu einem Dokument mit der Id docID. Die Daten in der Tabelle Document_Author werden mit den Daten des neuen AuthorHolders überschrieben. Dabei wird die Position entsprechend der Reihenfolge im AuthorHolder aktualisiert.

```
public static void updateDocumentAuthors(int docId,
                                         AuthorHolder authors)
throws InternalDbifError, InvalidIDException, IncompleteDataException
```

Parameters: docId - Die ID (PK) des Dokuments, dessen Urls aktualisiert werden sollen
authors - webshop.classes.AuthorHolder
Throws: InternalDbifError - Schwerer Fehler in webshop.dbif
InvalidIDException - Es existiert kein Dokument zum Parameter docId
IncompleteDataException - Der AuthorHolder ist leer

updateDocumentOffers Aktualisiert die Angebote zu einem Dokument mit der Id docID, zusätzliche Angebote werden in die Tabelle Offer eingefügt und die Angebote, deren Änderungskennzeichen gesetzt ist, werden in der Tabelle Offer (hier nur das valid_offer-Flag) aktualisiert.

```
public static void updateDocumentOffers(int docId,
                                        OfferHolder offers)
throws InternalDbifError, InvalidIDException, IncompleteDataException
```

Parameters: docId - Die ID (PK) des Dokuments, dessen Urls aktualisiert werden sollen
offers - webshop.classes.OfferHolder
Throws: InternalDbifError - Schwerer Fehler in webshop.dbif
InvalidIDException - Es existiert kein Dokument zum Parameter docId
IncompleteDataException - Der AuthorHolder ist leer

updateShopSettings Aktualisiert die Shop-Einstellungen. Falls noch kein Datensatz vorhanden ist, werden die Einstellungen in die Tabelle Shop_Settings eingefügt.

```
public static void updateShopSettings(ShopSettings settings)
throws InternalDbifError
```

Parameters: settings - webshop.classes.ShopSettings
Throws: InternalDbifError - Schwerer Fehler in webshop.dbif

insertAuthor Fügt einen neuen Eintrag in die Tabelle Author ein.

```
public static int insertAuthor(Author author)
throws InternalDbifError
```

Parameters: author - webshop.classesAuthor
Returns: Die vergebene ID (PK) für den neuen Autor
Throws: InternalDbifError - Schwerer Fehler in webshop.dbif

updateAuthor Aktualisiert einen Eintrag in der Tabelle Author.

```
public static void updateAuthor(Author author)
throws InternalDbifError, InvalidIDException
```

Parameters: author - webshop.classes.Author
Throws: InternalDbifError - Schwerer Fehler in webshop.dbif
InvalidIDException - Die ID des Autors ist ungültig

deleteAuthor Löscht einen Eintrag in der Tabelle Author.

```
public static void deleteAuthor(Author author)
throws InternalDbifError, InvalidIDException, InvalidDeleteIDException
```

Parameters: author - webshop.classes.Author
Throws: InternalDbifError - Schwerer Fehler in webshop.dbif
InvalidIDException - Die ID des Autors ist ungültig
InvalidDeleteIDException - Es existiert noch eine Fremdschlüsselbeziehung zu dem Datensatz

insertPublisher Fügt einen neuen Verlag in die Tabelle Publisher ein.

```
public static int insertPublisher(Publisher publisher)
throws InternalDbifError
```

Parameters: publisher - webshop.classes.Publisher
Returns: Die vergebene ID (PK) für den neuen Verlag
Throws: InternalDbifError - Schwerer Fehler in webshop.dbif

updatePublisher Aktualisiert einen Eintrag in der Tabelle Publisher.

```
public static void updatePublisher(Publisher publisher)
throws InternalDbifError, InvalidIDException
```

Parameters: publisher - webshop.classes.Publisher
Throws: InternalDbifError - Schwerer Fehler in webshop.dbif
InvalidIDException - Die ID des Publishers ist ungültig

deletePublisher Löscht einen Eintrag aus der Tabelle Publisher.

```
public static void deletePublisher(Publisher publisher)
throws InternalDbifError, InvalidIDException, InvalidDeleteIDException
```

Parameters: publisher - webshop.classes.Publisher
Throws: InternalDbifError - Schwerer Fehler in webshop.dbif
InvalidIDException - Die ID des Publishers ist ungültig

InvalidDeleteIDException - Es existiert noch eine Fremdschlüsselbeziehung zu dem Datensatz

insertDocumentType Fügt einen neuen Dokument-Typ in die Tabelle Document_Type ein.

```
public static int insertDocumentType(DocumentType docType)
throws InternalDbifError
```

Parameters: docType - webshop.classes.DocumentType
Returns: Die vergebene ID (PK) für den neuen Dokument-Typ
Throws: InternalDbifError - Schwerer Fehler in webshop.dbif

updateDocumentType Aktualisiert einen Eintrag in der Tabelle Document_Types.

```
public static void updateDocumentType(DocumentType docType)
throws InternalDbifError, InvalidIDException
```

Parameters: docTypes - webshop.classes.DocumentType
Throws: InternalDbifError - Schwerer Fehler in webshop.dbif
InvalidIDException - Die ID des Document-Typs ist ungültig

deleteDocumentType Löscht Eintrag aus der Tabelle Document_Type.

```
public static void deleteDocumentType(DocumentType docType)
throws InternalDbifError, InvalidIDException, InvalidDeleteIDException
```

Parameters: docType - webshop.classes.DocumentType
Throws: InternalDbifError - Schwerer Fehler in webshop.dbif
InvalidIDException - Die ID des Document-Typs ist ungültig
InvalidDeleteIDException - Es existiert noch eine Fremdschlüsselbeziehung zu dem Datensatz

insertLanguage Fügt eine neue Sprache in die Tabelle Language ein.

```
public static int insertLanguage(Language lang)
throws InternalDbifError
```

Parameters: lang - webshop.classes.Language
Returns: Die vergebene ID (PK) für die neue Sprache
Throws: InternalDbifError - Schwerer Fehler in webshop.dbif

updateLanguage Aktualisiert einen Eintrag in der Tabelle Language.

```
public static void updateLanguage(Language lang)
throws InternalDbifError, InvalidIDException
```

Parameters: lang - webshop.classes.Language
Throws: InternalDbifError - Schwerer Fehler in webshop.dbif
InvalidIDException - Die ID der Language ist ungültig

deleteLanguage Löscht Eintrag aus der Tabelle Language.

```
public static void deleteLanguage(Language lang)
throws InternalDbifError, InvalidIDException, InvalidDeleteIDException
```

Parameters: lang - webshop.classes.Language
Throws: InternalDbifError - Schwerer Fehler in webshop.dbif
InvalidIDException - Die ID der Language ist ungültig

InvalidDeleteIDException - Es existiert noch eine Fremdschlüsselbeziehung zu dem Datensatz

insertDocumentFormat Fügt ein neues Dokumenten-Format in die Tabelle Format ein.

```
public static int insertDocumentFormat(DocumentFormat format)
throws InternalDbifError
```

Parameters: format - webshop.classes.DocumentFormat
Returns: Die vergebene ID (PK) für das neue Format
Throws: InternalDbifError - Schwerer Fehler in webshop.dbif

updateDocumentFormat Aktualisiert einen Eintrag in der Tabelle Format.

```
public static void updateDocumentFormat(DocumentFormat format)
throws InternalDbifError, InvalidIDException
```

Parameters: format - webshop.classes.DocumentFormat
Throws: InternalDbifError - Schwerer Fehler in webshop.dbif
InvalidIDException - Die ID des Formats ist ungültig

deleteDocumentFormat Löscht einen Eintrag in der Tabelle Format.

```
public static void deleteDocumentFormat(DocumentFormat format)
throws InternalDbifError, InvalidIDException, InvalidDeleteIDException
```

Parameters: format - webshop.classes.DocumentFormat
Throws: InternalDbifError - Schwerer Fehler in webshop.dbif
InvalidIDException - Die ID des Formats ist ungültig
InvalidDeleteIDException - Es existiert noch eine Fremdschlüsselbeziehung zu dem Datensatz

insertFolder Fügt einen neuen Folder in die Tabelle Folder ein.

```
public static int insertFolder(Folder newFolder)
throws InternalDbifError, InvalidIDException
```

Parameters: Folder - webshop.classes.Folder
Returns: Die vergebene ID (PK) für den neuen Folder
Throws: InternalDbifError - Schwerer Fehler in webshop.dbif
InvalidIDException - Die FatherFolderId des Folders ist ungültig

deleteFolder Löscht einen Folder aus der Tabelle Folder. Seine Subfolder werden an seinen FatherFolder gehangen und die Dokumente innerhalb des Folders werden folderlos gesetzt, indem alle Zuordnungen in der Tabelle Document_Folder zu dem gelöschten Folder ebenfalls entfernt werden.

```
public static void deleteFolder(Folder obsoletFolder)
throws InternalDbifError, InvalidIDException, InvalidSemanticException
```

Parameters: obsoletFolder - webshop.classes.Folder
Throws: InternalDbifError - Schwerer Fehler in webshop.dbif
InvalidIDException - Es existiert kein Folder zur ID des Folders
InvalidSemanticException - Die FatherFolderId des Folders stimmt nicht mit dem Eintrag in der Tabelle Folder überein.

updateFolder Aktualisiert die Folder-Daten in Tabelle Folder.

```
public static void updateFolder(Folder updatedFolder)
throws InternalDbifError, InvalidIDException
```

Parameters: Folder - webshop.classes.Folder
Throws: InternalDbifError - Schwerer Fehler in webshop.dbif
InvalidIDException - Es existiert kein Folder zur ID des Folders

insertDocumentIntoFolder Fügt eine neue Zuordnung eines Dokuments zu einem Folder in die Tabelle Document_Folder der Datenbank ein.

```
public static void insertDocumentIntoFolder(int docId, int folderId)
throws InternalDbifError, InvalidIDException, InvalidWriteIDException
```

Parameters: docId - Die ID (PK) des Dokuments in der Tabelle Document
folderId - Die ID (PK) des Folders in der Tabelle Folder
Throws: InternalDbifError - Schwerer Fehler in webshop.dbif
InvalidIDException - Es existiert kein Folder zum Parameter folderId oder aber kein Dokument zum Parameter docId
InvalidWriteIDException - Das Dokument wurde bereits in den Folder eingefügt

deleteDocumentFromFolder Löscht die Zuordnung eines Dokuments zu einem Folder aus der Tabelle Document_Folder.

```
public static void deleteDocumentFromFolder(int docId, int folderId)
throws InternalDbifError, InvalidIDException
```

Parameters: docId - Die ID (PK) des Dokuments in der Tabelle Document
folderId - Die ID (PK) des Folders in der Tabelle Folder
Throws: InternalDbifError - Schwerer Fehler in webshop.dbif
InvalidIDException - Es existiert keine Zuordnung des Dokuments mit der docId zu dem Folder mit der folderId

getFolderlessDocuments Liefert alle Dokumente aus der Tabelle Document, die keinem Folder zugeordnet sind. Wenn keine Dokumente existieren, wird ein leerer Vector zurückgegeben. Die Dokumente enthalten nur die docId, den Titel und die Image-Url jedoch keine DocumentInfo.

```
public static java.util.Vector getFolderlessDocuments()
throws InternalDbifError
```

Returns: Vektor mit Objekten vom Typ webshop.classes.Document
Throws: InternalDbifError - Schwerer Fehler in webshop.dbif

getDocuments Liefert alle Dokumente aus der Tabelle Document, die dem Folder mit der ID im Parameter folderId zugeordnet sind. Wenn keine Dokumente existieren, wird ein leerer Vektor zurückgegeben. Die Dokumente enthalten nur die docId, den Titel und die Image-Url jedoch keine DocumentInfo.

```
public static java.util.Vector getDocuments(int folderId)
throws InternalDbifError, InvalidIDException
```

Parameters: folderId - Die ID (PK) in der Tabelle Folder
Returns: Vektor mit Objekten vom Typ webshop.classes.Document
Throws: InternalDbifError - Schwerer Fehler in webshop.dbif
InvalidIDException - Es existiert kein Folder zum Parameter folderId

3.2.3 Die Schnittstellen für den Payment-Handler

insertShop Fügt einen neuen Shop in die Tabelle Shop ein.

```
public static int insertShop(Login shop)
throws InternalDbifError, InvalidIDException
```

Parameters: shop - webshop.payment.Login
Returns: Die ID (PK) des Shops in der Tabelle Shop
Throws: InternalDbifError - Schwerer Fehler in webshop.dbif
InvalidIDException - Die ID des Shops ist ungültig

updateShop Ändert einen Shop Datensatz in der Tabelle Shop.

```
public static void updateShop(Login shop)
throws InternalDbifError, InvalidIDException
```

Parameters: shop - webshop.payment.Login
Throws: InternalDbifError - Schwerer Fehler in webshop.dbif
InvalidIDException - Die ID des Shops ist ungültig

getAllShops Liefert alle Shops aus der Tabelle Shop.

```
public static java.util.Vector getAllShops()
throws InternalDbifError
```

Returns: Vector mit Objekten vom Typ webshop.payment.Login
Throws: InternalDbifError - Schwerer Fehler in webshop.dbif

insertTransaction Fügt eine neue Transaktion zum Kunden eines Shops in die Tabelle Transaction ein.

```
public static int insertTransaction(Transaction transaction)
throws InternalDbifError, InvalidIDException
```

Parameters: transaction - webshop.payment.Transaction
Returns: Die ID (PK) der Transaktion in der Tabelle Transaction
Throws: InternalDbifError - Schwerer Fehler in webshop.dbif
InvalidIDException - Ungültige Fremdschlüsselbeziehung: Die CustomerId existiert nicht in der Tabelle Payment_Customer

updateTransaction Ändert eine Transaktion zum Kunden eines Shops in der Tabelle Transaction

```
public static void updateTransaction(Transaction transaction)
throws InternalDbifError, InvalidIDException
```

Parameters: transaction - webshop.payment.Transaction
Throws: InternalDbifError - Schwerer Fehler in webshop.dbif
InvalidIDException - Ungültige Fremdschlüsselbeziehung: Die CustomerId existiert nicht in der Tabelle Payment_Customer

getTransaction Liefert die Transaktion zu der angegebenen phPayId aus der Tabelle Transaction.

```
public static Transaction getTransaction(int phPayId)
throws InternalDbifError, InvalidIDException
```

Parameters: phPayId - die ID (PK) der Transaktion aus der Transaction-Tabelle
Returns: webshop.payment.Transaction
Throws: InternalDbifError - Schwerer Fehler in webshop.dbif
InvalidIDException - Es existiert keine Transaktion zum Parameter phPayId

getPaymentCustomerTransactions Liefert alle Transaktionen eines Kunden aus der Tabelle Transaction. Falls keine Transaktionen gespeichert sind, wird ein leerer Vektor zurückgegeben.

```
public static java.util.Vector
              getPaymentCustomerTransactions(int shop_customerId)
throws InternalDbifError, InvalidIDException
```

Parameters: shop_customerId - die ID (PK) des Kunden aus der Tabelle Payment_Customer
Returns: einen Vector mit Objekten vom Typ Transaction
Throws: InternalDbifError - Schwerer Fehler in webshop.dbif
InvalidIDException - Es existiert kein Kunde zum Parameter shop_customerId

insertPaymentCustomer Fügt einen neuen Kunden Datensatz in die Tabelle Payment_Customer ein.

```
public static int
              insertPaymentCustomer(PaymentCustomer paymentCustomer)
throws InternalDbifError, InvalidIDException
```

Parameters: paymentCustomer - webshop.payment.PaymentCustomer
Returns: Die ID (PK) des Kunden aus der Tabelle Payment_Customer
Throws: InternalDbifError - Schwerer Fehler in webshop.dbif
InvalidIDException - Ungültige Fremdschlüsselbeziehung: Die ShopId existiert nicht in der Tabelle Shop

updatePaymentCustomer Ändert einen Kunden-Datensatz in der Tabelle Payment_Customer.

```
public static void
       updatePaymentCustomer(PaymentCustomer paymentCustomer)
throws InternalDbifError, InvalidIDException
```

Parameters: paymentCustomer - webshop.payment.PaymentCustomer
Throws: InternalDbifError - Schwerer Fehler in webshop.dbif
InvalidIDException - Ungültige Fremdschlüsselbeziehung: Die ShopId existiert nicht in der Tabelle Shop

getAllPaymentCustomers Liefert alle Kunden aus der Tabelle Payment_Customer.

```
public static java.util.Vector getAllPaymentCustomers()
throws InternalDbifError
```

Returns: Vector mit Objekten vom Typ PaymentCustomer
Throws: InternalDbifError - Schwerer Fehler in webshop.dbif

getPaymentCustomer Liefert den Kunden mit der Id shop_customer_id aus der Tabelle Payment_Customer .

```
public static PaymentCustomer
              getPaymentCustomer(int shop_customer_id)
throws InternalDbifError, InvalidIDException
```

Parameters: shop_customer_id - die ID (PK) des Kunden aus der Payment_Customer-Tabelle
Returns: webshop.payment.PaymentCustomer
Throws: InternalDbifError - Schwerer Fehler in webshop.dbif
InvalidIDException - Es existiert kein Kunde zur angegebenen ID shop_customer_id in der Tabelle Payment_Customer

checkLogin Vergleicht das Login-Object mit der Shop-Tabelle in der Datenbank. Liefert true falls in der Tabelle Shop ein Datensatz mit Login.shopId = shop-tabelle.shop_id und Login.password = shop-tabelle.password existiert. Sonst false.

```
public static boolean checkLogin(Login login)
throws InternalDbifError, InvalidIDException
```

Parameters: Login - webshop.payment.Login
Returns: TRUE falls Login.shopId = shop-tabelle.shop_id und Login.password = shop-tabelle.password, sonst FALSE
Throws: InternalDbifError - Schwerer Fehler in webshop.dbif
InvalidIDException - Es existiert kein Shop zur shopId im Objekt Login

shopCustomerIdToCustomerId Liefert die shop_customer_id (PK) aus der Tabelle Payment_Customer zu der angegebenen customer_id und der shop_id aus dem Login-Objekt.

```
public static int
       shopCustomerIdToCustomerId(int customer_id, Login login)
throws InternalDbifError, InvalidIDException
```

Parameters: customer_id - die customer_id aus der Tabelle
Payment_Customer (entspricht der customer_id aus der Tabelle Customer
des Shops)
login - webshop.payment.Login *Returns:* shop_customer_id (PK) aus der Tabelle Payment_Customer
Throws: InternalDbifError - Schwerer Fehler in webshop.dbif
InvalidIDException - In der Tabelle Payment_Customer existiert kein Datensatz zu den angegebenen IDs

3.2.4 Ausnahmen- und Fehlerklassen

3.2.4.1 InternalDbifError

Diese Klasse behandelt den Fall, dass schnittstelleninterne Fehler auftreten. Darunter fallen insbesondere falsche SQL-Anweisungen, die durch fehlerhafte Programmierung auftreten. Hier ist die Verwendung von falschen Tabellen- oder Attributnamen zu nennen oder eine falsche Anzahl von Attributen sowie nicht übereinstimmende Typen. Die Klasse erweitert *java.lang.Error*.

Konstruktor: `public InternalDbifError(String msg, int type)`

Felder: `public final static int UNKNOWN_ERROR` - Unbekannte Fehlerursache
`public final static int DATABASE_ERROR` - Fehler in der Datenbank/Tabellen

Methoden: `public int getErrorType()`
- Liefert die Art des Fehlers (UNKNOWN_ERROR oder DATABASE_ERROR)

3.2.4.2 BadFitException

Dieser Fehler wird bei einer Fehlpassung von Url und SessionID ausgelöst.

Konstruktor: `public BadFitException(String msg, String url, String sessionId)`

Methoden: `public String getUrl()` - Liefert die Url der Anfrage
`public String getSessionID()` - Liefert die SessionID der Anfrage
`public String getMessage()` - Liefert eine Beschreibung des Fehlers

3.2.4.3 IncompleteDataException

Diese Klasse behandet den Fall, dass ein Dokument-Objekt mit unvollständigen Informationen in die Datenbank eingefügt wird.

Konstruktor: `public IncompleteDataException(String msg, int missingDataType)`

Felder: `public final static int UNKNOWN_DATA_TYPE` - Unbekannter Datentyp
`public final static int NO_OFFER_DATA` - Keine Angebotsdaten
`public final static int NO_AUTHOR_DATA` - Keine Autorendaten
`public final static int NO_SHOPURL_DATA` - Keine URLs
`public final static int NO_PUBLISHER_DATA` - Kein Verlag
`public final static int NO_DOCTYPE_DATA` - Keine Typangabe
`public final static int NO_FORMAT_DATA` - Keine Formatangabe
`public final static int NO_LANGUAGE_DATA` - Keine Sparachenangabe

Methoden: `public int getMissingDataType()` - Liefert den Typ der fehlenden Information
`public String getMessage()` - Liefert eine Beschreibung des Fehlers

3.2.4.4 InvalidDeleteIDException

Dieser Fehler wird ausgelöst, wenn ein Datensatz gelöscht werden soll, zu dem noch eine Fremdschlüsselbeziehung in der Datenbank besteht.

Konstruktoren: `public InvalidDeleteIDException(String msg, String siD)`
`public InvalidDeleteIDException(String msg, int iD)`

Methoden: `public String getSessionID()` - Liefert die Session-ID des zu löschenden Datensatzes
`public int getID()` - Liefert die ID des zu löschenden Datensatzes
`public String getMessage()` - Liefert eine Beschreibung des Fehlers

3.2.4.5 InvalidIDException

Dieser Fehler tritt auf, wenn versucht wird, mit einer unbekannten ID über den Primärschlüssel auf einen Datensatz zuzugreifen.

Konstruktoren: `public InvalidIDException(String msg, int iD)`
`public InvalidIDException(String msg, String sessionId)`

Methoden: `public String getSessionID()` - Liefert die ungültige Session-ID
`public int getID()` - Liefert die ungültige ID
`public String getMessage()` - Liefert eine Beschreibung des Fehlers

3.2.4.6 InvalidLoginException

Dieser Fehler wird bei einer Nutzerautentifizierung mit ungültiger Login-Passwort-Kombination ausgelöst.

Konstruktor: `public InvalidLoginException(String msg, String lgn)`

Methoden: `public String getLogin()` - Liefert das ungültige Login
`public String getMessage()` - Liefert eine Beschreibung des Fehlers

3.2.4.7 InvalidSemanticException

Dieser Fehler behandelt den Fall, dass ein Update, Insert oder Delete auf der Datenbank fehlschlägt, weil eine externe Regel verletzt wurde. Das kann bei doppeltem Einfügen von Daten der Fall sein, oder aber auch beim Löschen von Objekten, die nicht mit einem Datensatz in der Datenbank übereinstimmen.

Konstruktor: `public InvalidSemanticException(String msg)`

Methoden: `public String getMessage()` - Liefert eine Beschreibung des Fehlers

3.2.4.8 InvalidWriteIDException

Dieser Fehler behandelt die Verletzung der Eindeutigkeit des Primärschlüssels beim Eintragen eines neuen Datensatzes in eine Tabelle.

Konstruktoren: `public InvalidWriteIDException(String msg, int iD)`
`public InvalidWriteIDException(String msg, String sid)`

Methoden: `public String getSessionID()` - Liefert die ungültige Session-ID
`public int getID()` - Liefert die ungültige ID
`public String getMessage()` - Liefert eine Beschreibung des Fehlers

3.3 Realisierung

3.3.1 Zeitliche Vorgehensweise bei der Realisierung

Anforderungsanalyse und Design – Für die Realisierung der DDS-DB-Komponente wurden zunächst die erforderlichen ER-Modelle in Abstimmung mit den anderen Gruppen erstellt und in ein relationales Datenbankschema umgesetzt. Soweit es zu diesem Zeitpunkt möglich war, wurden bereits die Attribute zu den Relationen bestimmt. In dieser Phase wurde das Tool *ERWin (Platinum)* unter Windows NT eingesetzt, um schnelle Anpassungen an den ER-Modellen bzw. relationalen Schemata vornehmen zu können.
Anhand typischer Anwendungsfälle und Szenarios wurden unterschiedliche Methoden für den Zugriff auf die Datenbank identifiziert. In einem weiteren Schritt konnten einige ähnliche Methoden, die von unterschiedlichen DDS-Komponenten benötigt wurden, so in Übereinstimmung gebracht werden, dass eine mehrfache Implementierung vermieden werden konnte.
Mögliche Fehler, die beim Zugriff auf die Datenbank auftreten können, werden durch speziell implementiert Java-Fehler- bzw. Ausnahme-Klassen behandelt. Die Identifikation dieser Fehler fand zum größten Teil erst während der Implementierung statt.

Ergebnis der Designphase ist die Umsetzung der DDS-DB-Komponente durch im wesentlichen zwei Java-Klassen, eingebunden in das Paket *webshop.dbif*. Die eigentliche Schnittstelle bildet die Klasse *DB_Access*. Diese benutzt die Klasse *DBMediator*. Der DBMediator verwaltet einen Pool von freien Connection-Objekten (Verbindungen zur Datenbank) in Form eines Stapels (Stack). Dieses Konzept wurde aus dem *eVerlage-System* des OFFIS übernommen. Der Vorteil der Verwendung eines solchen Mediators liegt hauptsächlich in einem Geschwindigkeitsgewinn. Der Aufbau einer Datenbankverbindung beansprucht u.U. einige Zeit. Daher ist es ungünstig für jede Anfrage oder Transaktion eine eigene Verbindung aufzubauen. Der DBMediator löst dieses Problem dadurch, dass er bereits bei seiner Instanziierung eine Anzahl an Datenbankverbindungen öffnet. Eine Anwendung oder ein Thread, in unserem Fall eine Methode aus der DB-Schnittstelle *DB_Access*, kann dann bei Bedarf eine bereits geöffnete Verbindung anfordern und nach Beendigung seiner Anfrage wieder freigeben. Steht keine freie Verbindung mehr zur Verfügung, wird der Anfragende Prozess solange angehalten.

Implementierung – Als Entwicklungsplattform kamen die *Sun* Workstations des OFFIS-Softwarelabors mit dem Betriebssystem Solaris sowie das JDK 1.2 von *Sun* zum Einsatz. Als DBMS wurde Oracle genutzt, auf herstellerspezifische Erweiterungen wurde dabei aber verzichtet. Ein entsprechender JDBC-Treiber war bereits vorhanden (Version 8.1.6). Alle datenbankspezifischen Einstellungen und Parameter, die für den Aufbau einer Datenbankverbindung notwendig sind, werden in einer dafür vorgesehenen zentralen Datei *webshop.ini* (siehe Installation) vorgenommen und dort vom DBMediator ausgelesen.

In der Implementierungsphase wurde zunächst eine „Dummy"-Schnittstelle geschaffen, deren Methoden noch nicht auf die Datenbank zugreifen konnten. Die vollständige Implementierung und Test der einzelnen Methoden erfolgte in der Reihenfolge ihrer Priorität für die jeweiligen Teilgruppen, die für die Umsetzung der anderen DDS-Komponenten zuständig waren.
Während der Implementierung der einzelnen Methoden der Schnittstelle wurden weitere

Fehlerfälle und Ausnahmen identifiziert. Entsprechend wurden die Klassen für die Behandlung der Ausnahmen Implementiert, so dass anschließend eine vollständige Deklaration der Methoden inklusive der Implementierung und Test in einer Testumgebung erfolgen konnte. Nach erfolgreichem Abschluss aller Testfälle wurde die „Dummy"-Methode durch die Implementierung ersetzt.

Integration – In der Integrationsphase wurde die Zusammenarbeit zwischen DDS-DB-Schnittstelle und den anderen DDS-Komponenten getestet. Dabei auftretende Fehler resultierten aus bisher ungetesteten Szenarios und konnten beseitigt werden. Außerdem stellte sich während der Integration heraus, dass eine Reihe von neuen Anforderungen an die Schnittstelle auftauchten, die dann erst in der Schlussphase umgesetzt wurden. Hier seien Modifikationen an der Tabelle *Shop_Settings*, die Erweiterung der Suchfunktionalität sowie die Ausgliederung und Umstrukturierung der Entitäten für den Payment-Handler genannt.

3.3.2 Aufbau der DDS-DB-Komponente

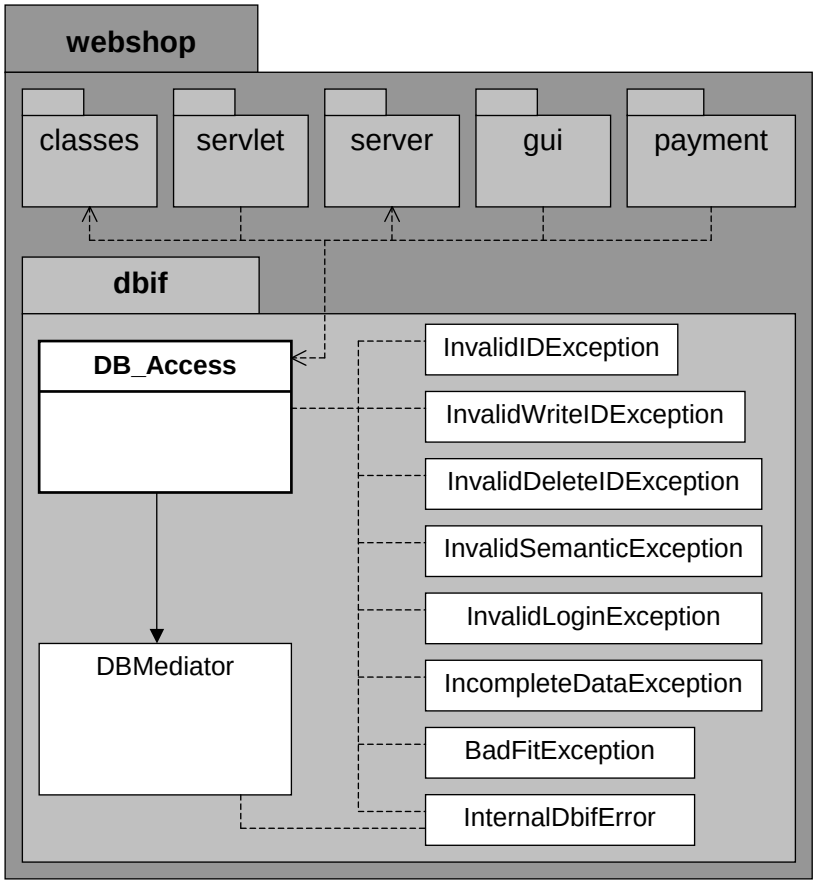

Abbildung 3.4: webshop.dbif - Paketstruktur

3.4 DB-Optimierungsaspekte

Bei der gegenwärtigen Implementierung der Datenbank-Komponente ging es zunächst darum, die von den jeweiligen Teilsystemen geforderte Funktionalität innerhalb der gesetzten Zeit und Dauer der Projektgruppe in den Schnittstellen umzusetzen. Es bleibt auf jedem Niveau der Implementierung immer noch ein gewisser Spielraum, um die Performance der Datenbankzugriffe zu verbessern bzw. weiter zu optimieren.
Die zum Abschluss der Projektgruppe vorliegende Version der DB-Schnittstellen beschränkt sich auf die Konzepte, die durch das JDBC1.0 API bereitgestellt werden. Sämtliche Statements, über die die Datenbank kontaktiert wird, verwenden Standard-SQL Anfragen. Dies erlaubt dem Nutzer des gesamten DDS-Systems eine größere Flexibilität bei der Wahl des DBMS, mit dem er seinen Shop aufbauen möchte.
Die Optimierungsmöglichkeiten, bei denen dieser Vorteil aufrecht erhalten bleibt, sind z.B. folgende:

Verwendung spezieller Sichten (Views) innerhalb der Datenbank – Durch den Einsatz von Views kann die wiederholte Ausführung von Joins vermieden werden, solange sich die Informationen in den zugrunde liegenden Tabellen nicht ändern. Views bieten sich für die DDS-DB vor allem bei solchen Tabellen an, die lediglich durch die DDS-Authoring-Komponente verändert werden.

Optimierung von Joins durch den Einsatz von Speziellen Datenbank-Tools – Joins, die sich über eine größere Anzahl von Tabellen erstrecken, hängen in ihrer Ausführungsgeschwindigkeit sehr stark von der Reihenfolge der Verkettung der Tabellen ab. Mit geeigneten Tools lässt sich diese Geschwindigkeit für unterschliedliche Kombinationen auswerten, um einen optimalen SQL-Ausdruck zu finden.

Indizierung von Attributen – Durch die Generierung geeigneter Indexe (physische Datenstrukturen) kann die Zugriffsgeschwindigkeit generell verbessert werden. Insbesondere sind hier Fremdschlüssel und Attribute, die als Auswahlkriterium für Joins dienen, zu nennen.

Verwendung von Prepared Statements – Wird oftmals hintereinander mit dem gleichen SQL-Ausdruck, bei dem sich lediglich die eingesetzten Werte verändern, auf die Datenbank zugegriffen, bietet sich unter Java ein *Prepared Statement* an. Da das Statement bereits vorcompiliert ist, und nur noch die veränderlichen Werte eingesetzt werden müssen, kann so ein Geschwindigkeitsgewinn erreicht werden. Diese Methode bietet sich z.B. besonders in den DDS-DB-Schnittstellen an, über die neue Dokumente in die Datenbank eingetragen bzw. aktualisiert werden, da u.U. eine große Anzahl an URLs nacheinander eingetragen werden müssen.

Neben den oben genannten Möglichkeiten, ließen sich evtl. auch proprietäre Erweiterungen des Herstellers des Datenbanksystems nutzen. Dies geht allerdings auf Kosten der Kompatibilität zu anderen Datenbank-Systemen. Es kann auch zu Problemen bei der Verwendung unterschiedlicher Treiberversionen kommen. Der JDBC-Treiber des verwendeten Systems (*Oracle8i JDBC Release 8.1.5 bzw. 8.1.6*) bietet unter anderem die folgenden beiden interessanten Möglichkeiten:

Batch Updates – Durch so genannte *Batch Updates* kann die Anzahl der Kontaktaufnahmen zum Datenbankserver verringert werden. Es werden mehrere Statements zusammengefasst

und zu einem wählbaren Zeitpunkt in einem Paket bzw. als Batch an die Datenbank zur Ausführung gesendet.

Stored Procedures – Die auf dem Datenbankserver gespeicherten Prozeduren in JAVA oder PLSQL bieten eine optimale Anpassung an das DBMS, sind aber bei Veränderungen, Wartung und Erweiterung des Systems am wenigsten flexibel. Ihr Einsatz lohnt sich insbesondere bei komplexeren Transaktionen, da hier der anteilig größte Geschwindigkeitsgewinn zu erwarten ist.

Kapitel 4

Payment - Komponente

4.1 Entwurf

Eine Teilaufgabe bei der Entwicklung des DDS-Systems besteht darin, die Bezahlung der digitalen Güter zu realisieren. Das DDS ist zum Anbieten von kostenpflichtigen Web-Seiten vorgesehen. Um die kostenpflichtigen Web-Seiten zu besuchen, muss der Nutzer für die gewünschten Inhalte Zeit-Lizenzen erwerben. Dieser Ablauf macht es notwendig, dass die Bezahlung vor dem Aufruf der kostenpflichtigen Seiten erfolgt. Bezahlungen per Überweisung bzw. Kreditkarte eignen sich hierbei weniger gut, da bis zum Eingang der Zahlung u.U. mehrere Tage vergehen können. Eher geeignet sind Zahlungsverfahren, die eine umgehende Zahlung ohne Zeitverlust ermöglichen. Diesen Vorteil bieten elektronische Zahlungsverfahren. Damit besteht die Möglichkeit, für eine gewünschte Web-Seite sofort eine Zeit-Lizenz, durch Übermittlung des elektronischen Geldes (z.B. elektronische Münzen), zu erwerben. Hierbei entstehen nur geringe Zeitverluste im Sekundenbereich, bis der Bezahlvorgang abgeschlossen ist. Der Nutzer kann dann unmittelbar danach seine gewünschte Web-Seite aufrufen.

Die Bezahlungen für kostenplichtige Web-Seiten soll durch einen Bezahl-Server (im weiteren PaymentHandler) verwaltet werden. Der PaymentHandler wird bei Bestellung von kostenpflichtigen Web-Seiten kontaktiert und generiert, je nach gewähltem Zahlungsverfahren, eine Zahlungsaufforderung für den Kunden. Die Zahlungsaufforderung wird in einer Datenbank gespeichert. Beim Bezahlen durch Kunden nimmt der PaymentHandler die Zahlung entgegen und verifiziert diese. Ist der Bezahlvorgang erfolgreich, wird der in der Datenbank gespeicherte Vorgang (Transaktion) als bezahlt gekennzeichnet. In welcher Form der PaymentHandler die Zahlung entgegen nimmt, hängt von dem gewählten Zahlungssystem des Kunden ab.

Der PaymentHandler soll mehrere DDS verwalten können. Hierfür läuft der PaymentHandler als Server auf einem separaten Host. Ein DDS arbeitet als Client des PaymentHandler und kann dessen Dienste nach erfolgreicher Authentifizierung nutzen. Zur Realisierung des Client/Server-Komponente dient die RMI-Technologie aus dem Sun JDK.

Beispielhaft werden die Zahlungsverfahren Kundenkonto (Account), Banküberweisung, Kreditkarte, ECash und CyberCash implementiert. Bei CyberCash und ECash handelt es sich um sogenannte elektronische Zahlungsverfahren, die eine unmittelbare Übertragung von Geldbeträgen ermöglichen. Das Kundenkonto ist kein Zahlungsverfahren im eigentlichen Sinne. Das Kundenkonto bietet sich an, wenn kein elektronisches Zahlungsverfahren zur Verfügung steht.

Es kann in dem Fall mittels eines herkömmlichen Zahlungsverfahrens (z.B. Banküberweisung) ein Geldbetrag auf das Kundenkonto übertragen werden. Sobald der Geldbetrag auf das Kundenkonto gebucht ist, kann es zur Bezahlung benutzt werden.

In Ermangelung eines HBCI-API für das Betriebssystem UNIX haben wir uns entschlossen, das Buchen der Zahlungen per Banküberweisung/Kreditkarte manuell vorzunehmen. Es wird ein Administrations-Tool geschaffen, das die Möglichkeit bietet, die Transaktion des Payment-Handler anzuzeigen und Transaktionen, die als offen gekennzeichnet sind, zu buchen. Langfristig empfiehlt es sich jedoch, HBCI zu nutzen, was eine automatische Prüfung der Zahlungseingänge ermöglicht. Ansonsten bleibt nur der Weg zur Bank, um die Kontoauszüge zu prüfen.

4.2 Schnittstellen

4.2.1 Allgemeines

Die Komponenten des DDS nutzen den PaymentHandler mit Hilfe der Klasse *RMIClient*. Die Klasse *RMIClient* enthält alle im *PaymentHandlerInterface* beschriebenen Methoden (siehe unten). Zusätzlich kapselt die Klasse *RMIClient* alle RMI-relevanten Funktionen. Beim Erzeugen einer Instanz des *RMIClient* werden automatisch die für RMI nötigen Sicherheitseinstellungen geladen und die Verbindung zum PaymentHandler hergestellt. Der Übergabeparameter *Login* aus dem *PaymentHandlerInterface* wird für den *RMIClient* nicht benötigt, da der *RMIClient* diese Daten aus einer Konfigurationsdatei lädt (siehe auch Handbuch Kap 4.4.1), und bei Methodenaufrufen selbst einfügt.

4.2.2 PaymentHandlerInterface

- Vector getBrandIds ()
  - *Beschreibung:* Liefert die unterstützten Zahlungssysteme
  - *Rückgabewert:* Ein Vektor mit allen Zahlungssystemen (Brand)
  - *Ausnahme RemoteException:* Ein Fehler in der RMI-Verbindung
- Transaction getTransaction (int phPayId, Login login)
  - *Beschreibung:* Liefert einen gespeicherten Bezahlvorgang (Transaktion)
  - *Parameter phPayId:* Die Nummer der Transaktion, die zurückgegeben werden soll
  - *Parameter login:* Die Login-Daten des Shops (Username, Passwort)
  - *Rückgabewert:* Die Transaktion zur phPayId
  - *Ausnahme RemoteException:* Ein Fehler in der RMI-Verbindung
  - *Ausnahme IllegalParameterException:* Übergabeparameter sind inkonsistent
  - *Ausnahme TransactionDoesNotExistException:* Keine Transaktion zur phPayId vorhanden
  - *Ausnahme InternalDbifError:* Datenbankfehler
- Vector getCustomerTransactionsSinceDate (int customerId, Date date, Login login)

  - *Beschreibung:* Liefert für einen Kunden die vorhandenen Transaktionen ab einem bestimmten Datum
  - *Parameter customerId:* Identifikationsnummer des Kunden
  - *Parameter date:* Anfangsdatum der Transaktionen
  - *Parameter login:* Die Login-Daten des Shops
  - *Rückgabewert:* Ein Vektor mit Transaktionen
  - *Ausnahme RemoteException:* Ein Fehler in der RMI-Verbindung
  - *Ausnahme IllegalParameterException:* Übergabeparameter sind inkonsistent
  - *Ausnahme CustomerDoesNotExistException:* Kunde ist nicht vorhanden
  - *Ausnahme InternalDbifError:* Datenbankfehler

- int getTaStatus (int phPayId, Login login)
  - *Beschreibung:* Liefert den Status einer Transaktion
  - *Parameter phPayId:* Identifaktionsnummer der zu untersuchenden Transaktion
  - *Parameter login:* Die Login-Daten des Shops
  - *Rückgabewert:* ist 1, wenn es sich um einen Zahlungsaufforderung handelt, 2 bei einer gebuchten Zahlung
  - *Ausnahme RemoteException:* Ein Fehler in der RMI-Verbindung
  - *Ausnahme IllegalParameterException:* Übergabeparameter sind inkonsistent
  - *Ausnahme TransactionDoesNotExistException:* Keine Transaktion zur phPayId vorhanden
  - *Ausnahme InternalDbifError:* Datenbankfehler

- int getErrorStatus (int phPayId, Login login)
  - *Beschreibung:* Liefert den Fehlerstatus einer Transaktion
  - *Parameter phPayId:* Identifikationsnummer der zu untersuchenden Transaktion
  - *Parameter login:* Die Login-Daten des Shops
  - *Rückgabewert:* ist 0, wenn kein Fehler vorlag, ungleich 0 sonst
  - *Ausnahme RemoteException:* Ein Fehler in der RMI-Verbindung
  - *Ausnahme IllegalParameterException:* Übergabeparameter sind inkonsistent
  - *Ausnahme TransactionDoesNotExistException:* Keine Transaktion zur phPayId vorhanden
  - *Ausnahme InternalDbifError:* Datenbankfehler

- double getBalance (int customerId, Login login)
  - *Beschreibung:* Liefert zu einem Kunden den Kontostand
  - *Parameter customerId:* Identifaktionsnummer des Kunden
  - *Parameter login:* Die Login-Daten des Shops
  - *Rückgabewert:* Der Betrag des Kontostands
  - *Ausnahme RemoteException:* Ein Fehler in der RMI-Verbindung

  - *Ausnahme IllegalParameterException:* Übergabeparameter sind inkonsistent
  - *Ausnahme CustomerDoesNotExistException:* Kunde ist nicht vorhanden
  - *Ausnahme InternalDbifError:* Datenbankfehler

- double getCredit (int customerId, Login login)
  - *Beschreibung:* Liefert zu einem Kunden den gewährten Kreditbetrag
  - *Parameter customerId:* Identifaktionsnummer des Kunden
  - *Parameter login:* Die Login-Daten des Shops
  - *Rückgabewert:* Der Kreditbetrag
  - *Ausnahme RemoteException:* Ein Fehler in der RMI-Verbindung
  - *Ausnahme IllegalParameterException:* Übergabeparameter sind inkonsistent
  - *Ausnahme CustomerDoesNotExistException:* Kunde ist nicht vorhanden
  - *Ausnahme InternalDbifError:* Datenbankfehler

- InvoiceResult invoiceRequest (int customerId, String cOrgId, String shortDesc, int brandId, double amount, Date dateOfRequest, int transactionType, Login login)
  - *Beschreibung:* Erzeugt eine Zahlungsaufforderung für einen Kunden.
  - *Parameter customerId:* Identifaktionsnummer des Kunden
  - *Parameter cOrgId:* Nähere Informationen zum Kunden (z.B. Kontodaten)
  - *Parameter shortDesc:* Eine kurze Beschreibung der Zahlungsaufforderung
  - *Parameter brandId:* Das verwendete Zahlungssystem
  - *Parameter amount:* Betrag der Zahlungsauffoerderung
  - *Parameter dateOfRequest:* Datum der Zahlungsaufforderung
  - *Parameter transactionType:* Art der Zahlungsauffoerderung (1 = es wird etwas gekauft, 2 = Einzahlung auf das Kundenkonto)
  - *Parameter login:* Die Login-Daten des Shops
  - *Rückgabewert:* In Abhängigkeit vom gewählten Zahlungssystem wird ein Ergebnis (InvoiceResult) zurückgeliefert.
  - *Ausnahme RemoteException:* Ein Fehler in der RMI-Verbindung
  - *Ausnahme IllegalParameterException:* Übergabeparameter sind inkonsistent
  - *Ausnahme CustomerDoesNotExistException:* Kunde ist nicht vorhanden
  - *Ausnahme PaymentFailedException:* Bezahlvorgang konnte nicht durchgeführt werden
  - *Ausnahme InternalDbifError:* Datenbankfehler

- boolean transactionSettled (int phPayId, Login login)
  - *Beschreibung:* Informiert, ob ein Bezahlvorgang (Transaktion) abgeschlossen ist
  - *Parameter phPayId:* Identifaktionsnummer der Transaktion
  - *Parameter login:* Die Login-Daten des Shops
  - *Rückgabewert:* true, wenn die Bezahlung gebucht ist, sonst false

  - *Ausnahme RemoteException:* Ein Fehler in der RMI-Verbindung
  - *Ausnahme IllegalParameterException:* Übergabeparameter sind inkonsistent
  - *Ausnahme TransactionDoesNotExistException:* Keine Transaktion zur phPayId vorhanden
  - *Ausnahme InternalDbifError:* Datenbankfehler

- void insertPaymentCustomer (int customerId, Login login)
  - *Beschreibung:* Fügt einen neuen Kunden in das Bezahlsystem ein
  - *Parameter phPayId:* Identifaktionsnummer des Kunden
  - *Parameter login:* Die Login-Daten des Shops
  - *Ausnahme RemoteException:* Ein Fehler in der RMI-Verbindung
  - *Ausnahme IllegalParameterException:* Übergabeparameter sind inkonsistent
  - *Ausnahme InternalDbifError:* Datenbankfehler

- boolean bookPayment (int phPayId, Date paidAt, double amount, Login login)
  - *Beschreibung:* Eine Bezahlung wird gebucht
  - *Parameter phPayId:* Identifaktionsnummer der Transaktion
  - *Parameter paidAt:* Datum der Buchung
  - *Parameter amount:* Buchungsbetrag
  - *Parameter login:* Die Login-Daten des Shops
  - *Rückgabewert:* true, wenn die Bezahlung gebucht ist, sonst false
  - *Ausnahme RemoteException:* Ein Fehler in der RMI-Verbindung
  - *Ausnahme IllegalParameterException:* Übergabeparameter sind inkonsistent
  - *Ausnahme TransactionDoesNotExistException:* Keine Transaktion zur phPayId vorhanden
  - *Ausnahme InternalDbifError:* Datenbankfehler

4.2.3 Brand

- Konstanten:
  - int ACCOUNT = 1
  - int BANKTRANSFER = 2
  - int CREDITCARD = 3
  - int ECASH = 4
  - int CYBERCASH = 5
  - String ACCOUNT_STR = “Account“
  - String BANKTRANSFER_STR = “BankTransfer“
  - String CREDITCARD_STR = “CreditCard“
  - String ECASH_STR = “ECash“

  - String CYBERCASH_STR = “CyberCash“

- Attribute:
  - String brandName
  - int brandId
- Konstruktoren:
  - public Brand()
  - public Brand(int id)

4.2.4 InvoiceResult

- Attribute:
  - String message
  - String invoiceASCII
  - int phPayId
  - int error
- Konstruktoren:
  - public InvoiceResult()
  - public InvoiceResult(int phPayId, String message, String invoiceASCII, int error)
  - public InvoiceResult(int phPayId, String message, int error)

4.2.5 Login

- Attribute:
  - int shop_id
  - String password
  - String name
  - String bankConnection
- Konstruktoren:
  - public Login()
  - public Login(int shop_id, String password)
  - public Login(int shop_id, String password, String name, String connection)

4.2.6 Transaction

- Konstanten:
  - int TYPE_BUY = 1
  - int TYPE_LOAD = 2
  - int STATE_NEW = 1
  - int STATE_BOOKED = 2
- Attribute:
  - int phPayId
  - int taStatus
  - int errorStatus
  - String cOrgId
  - int customer_id
  - String shortDesc
  - int brandId
  - int transactionType
  - Date invoicedAt
  - double invoicedAmount
  - Date paidAt
  - double paidAmount
  - Date settledAt
  - int shopId
- Konstruktoren:
  - public Transaction()
  - public Transaction(int phPayId, int taStatus, int errorStatus, String cOrgId, int customer_id, String shortDesc, int brandId, int transactionType, Date invoicedAt, double invoicedAmount, Date paidAt, double paidAmount, Date settledAt, int shopId)
- spezielle Methoden:
  - public boolean isSettled()

4.3 Architektur und Implementierung des Payment-Systems

In diesem Kapitel sollen zunächst die für das Payment-System implementierten Klassen sowie die Beziehungen zwischen diesen vorgestellt werden. Anschließend wird die Implementierung der einzelnen Zahlungsverfahren erläutert.

4.3.1 Klassen des Payment-Systems

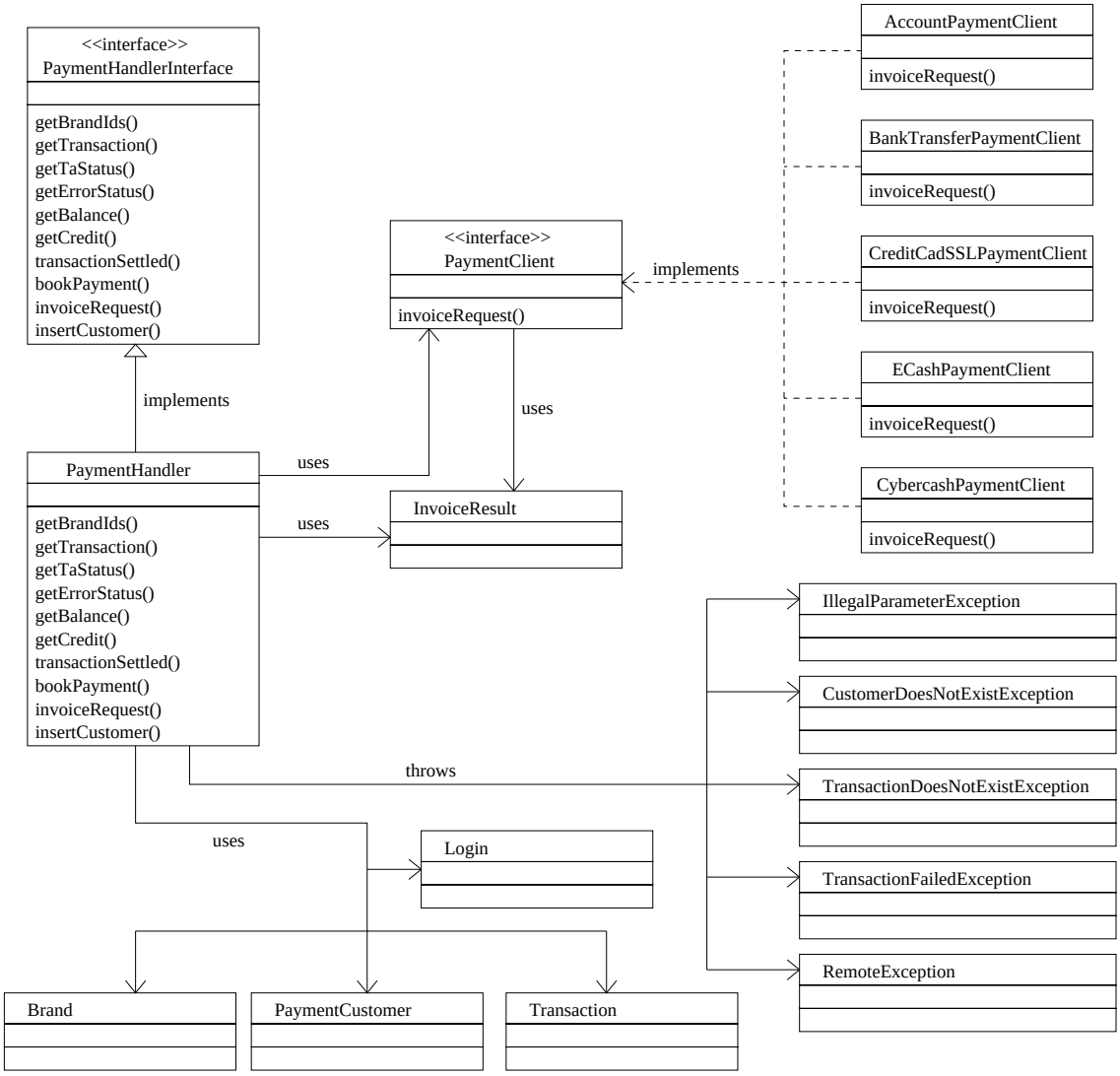

Abbildung 4.1: Klassenhierarchie des Payment-Systems

4.3.1.1 PaymentHandlerInterface und PaymentHandler

Die eigentliche, nach aussen sichtbare Schnittstelle des Payment-Systems wird durch die RMI-Schnittstellen-Klasse *PaymentHandlerInterface* definiert. Implementiert wird die gesamte Funktionalität dieser abstrakten Klasse durch *PaymentHandler*. Die Anwendung der Methoden wurde bereits im vorhergehenden Kapitel erläutert. Hier sollen nun einige Einzelheiten zur Implementierung folgen:

getBrandIds() :
Die Id's der implementierten Zahlungsverfahren werden hier in ein Vector-Objekt geschrieben und zurückgegeben.

getTransaction() :
Die angeforderte Transaktion wird über die Datenbank-Schnittstelle aus der Datenbank gelesen und als Transaction-Object zurückgegeben.

getTaStatus() :
Aus der Datenbank wird die vollständige Transaktion gelesen. Anschließend wird das Attribut *taStatus* daraus zurückgegeben.

getErrorStatus() :
Über die Datenbank-Schnittstelle wird die vollständige Transaktion ausgelesen. Zurückgegeben wird das Attribut *errorStatus*.

getBalance() :
Der Eintrag aus der Datenbank-Tabelle PaymentCustomer, dessen Attribut *shopCustomerId* mit dem der Methode getBalance() übergebenen Wert übereinstimmt, wird gelesen. Aus diesem wird das Attribut *balance* zurückgegeben.

getCredit() :
Der Eintrag aus der Datenbank-Tabelle PaymentCustomer, dessen Attribut *shopCustomerId* mit dem der Methode getCredit() übergebenen Wert übereinstimmt, wird gelesen. Aus diesem wird das Attribut *credit* zurückgegeben.

transactionSettled() :
Aus der Datenbank wird die vollständige Transaktion gelesen. Anschließend wird *true* zurückgegeben, falls das Attribut *taStatus* den Wert 2 hat, andernfalls *false*.

bookPayment() :
Über die Datenbank-Schnittstelle wird die Transaktion, die gebucht werden soll, eingelesen. Dann wird deren *taStatus* auf 2 (=gebucht) gesetzt. In *paidAt* und *settledAt* wird das aktuelle Datum eingetragen; das Attribut *paidAmount* wird auf den geforderten Betrag gesetzt. Anschliessend wird die somit gebuchte Transaktion in die Datenbank zurückgeschrieben.

invoiceRequest() :
Die Methode *invoiceRequest()* legt über die DB-Schnittstelle eine neue Transaktion mit den übergebenen Daten an. Anschließend wird je nach übergebener *brandID* eine Instanz des entsprechenden PaymentClients erzeugt, der das jeweilige Zahlungsverfahren realisiert. Für diesen PaymentClient wird wiederum die Implementierung der Methode *invoiceRequest()* aufgerufen, wodurch die Zahlung eingeleitet wird.
Treten hierbei Fehler auf, so wird eine entsprechende *PaymentFailedException* erzeugt. Bei einem fehlerfreien Zahlungsvorgang wird getestet, ob als Zahlungstyp (*transactionType* die Einzahlung auf ein Kundenkonto gewählt wurde. In diesem Fall muss getestet werden, ob der Zahlungsvorgang bereits abgeschlossen wurde. Ist dies der Fall, so wird der eingezahlte Betrag auf das Kundenkonto gutgeschrieben. Rückgabewert dieser Methode ist ein Objekt vom Typ *InvoiceResult*, das detaillierte Ergebnisse der Zahlungsaufforderung enthält.

insertCustomer() :
Diese Methode schreibt über die Datenbank-Schnittstelle einen neuen PaymentCustomer in die entsprechende Tabelle. *shopId* und *shopCustomerId* werden dabei auf die übergebenen Werte gesetzt, *balance* und *credit* werden mit Null initialisiert.

4.3.1.2 PaymentClient

Die abstrakte Klasse *PaymentClient* ist die Vaterklasse für alle konkreten Implementierungen der verschiedenen Zahlungsverfahren. Sie wird von *PaymentHandler* in der Methode *invoiceRequest()* deklariert und - abhängig vom gewählten Zahlungsverfahren - mit dem entsprechenden Zahlungssystem-PaymentClient instanziiert. Als einzige Methode wird hier *invoiceRequest()* deklariert. Diese wird von allen konkreten Zahlungssystem-Clients implementiert.

4.3.1.3 InvoiceResult

InvoiceResult stellt das Ergebnis einer Zahlungsaufforderung dar. Die zugehörige Transaktion ist durch das Attribut *phPayId* beschrieben. Das Attribut *error* enthält entweder den Wert Null oder einen Fehlercode, falls es während der Zahlung zu einem Fehler gekommen ist. *message* kann für einen zusätzlichen Fehlertext oder eine beliebige Nachricht verwendet werden. *invoiceASCII* wird benutzt, um für die elektronischen Zahlungsverfahren ECash und Cybercash die von den jeweiligen Zahlungsservern erzeugten Zahlungsaufforderungen, die an den Rechner des Kunden geschickt werden müssen, zu übergeben.

4.3.1.4 Login

Die Klasse *Login* wird benötigt, um den Händler, der die Methoden des Payment-Systems nutzen will, eindeutig zu identifizieren und zu authentifizieren. Dazu verfügt sie über die Attribute *userId* und *password*. Zur weiteren Information sind - wie in der entsprechenden Tabelle der Datenbank - die Attribute *name* und *bankConnection* vorhanden.

4.3.1.5 Transaction

Transaction repräsentiert eine Zahlung und besitzt folgende Attribute: *phPayId, brandId, customer_id, shopId, taStatus, errorStatus, cOrgId, shortDesc, invoicedAt, invoicedAmount, paidAt, paidAmount* und *settledAt*.

4.3.1.6 PaymentCustomer

Die Klasse *PaymentCustomer* stellt einen Kunden des Payment-Systems dar. Wie in der entsprechenden Datenbank-Tabelle sind die Attribute *customerId, balance, credit, shopId* und *shopCustomerId* vorhanden.

4.3.1.7 Brand

Brand identifiziert ein Zahlungssystem durch die ID (*brandId*) und den jeweiligen Namen (*brandName*).

4.3.1.8 IllegalParameterException, CustomerDoesNotExistException, TransactionDoesNotExistException, TransactionFailedException

Diese Exceptions sind von der Klasse *Exception* abgeleitet und besitzen nur die geerbten Attribute und Methoden.

4.3.2 Implementierung der Clients für die Zahlungsverfahren

Alle von der abstrakten Klasse *PaymentClient* abgeleiteten konkreten Zahlungsverfahren-Clients implementieren die Methode *invoiceRequest()*. Die vom jeweiligen Zahlungsverfahren abhängige Implementierung soll im folgenden beschrieben werden.

4.3.2.1 AccountPaymentClient

Die Klasse *AccountPaymentClient* realisiert die Zahlung über ein händlerspezifisches Kundenkonto.
In der Methode *invoiceRequest()* wird die zuvor vom *PaymentHandler* neu angelegte Transaktion aus der Datenbank ausgelesen; anschließend werden hieraus Kontostand und möglicher Überziehungskredit gelesen. Decken Kontostand und Kredit den zu zahlenden Betrag nicht ab, so wird ein InvoiceResult-Objekt mit der entsprechenden Nachricht (NOT_ENOUGH_MONEY) zurückgegeben. Ist die Summe ausreichend, so wird der Betrag vom Kundenkonto abgezogen. Um die Transaktion zu buchen, werden die Attribute *paidAt, paidAmount, taStatus* und *settledAt* gesetzt. Dann werden Kundendaten und Transaktion über die Datenbankschnittstelle in der Datenbank aktualisiert. Zurückgegeben wird ein InvoiceResult-Objekt ohne Fehler (NO_ERROR).

4.3.2.2 BankTransferPaymentClient

BankTransferPaymentClient ist für Zahlungen per Bankeinzug zuständig. Da das Payment-System jedoch keine entsprechende Schnittstelle integriert, um diesen Zahlungsvorgang zu automatisieren (bspw. HBCI), führt die Methode *invoiceRequest* hier keine weiteren Aktionen durch. Das einer offenen Zahlungsaufforderderung entsprechende Transaktionsobjekt wird bereits im Aufruf des *PaymentHandlers* erzeugt. Bei erfolgtem Bankeinzug kann die Buchung dann über das *TransactionTool* erfolgen.

4.3.2.3 CreditCardSSLPaymentClient

CreditCardSSLPaymentClient kann für Zahlungen per Kreditkarte benutzt werden. Allerdings fehlt auch hier die Automatisierung des Zahlungsvorganges. Dieser muss durch den Administrator des PaymentHandlers initiiert werden und kann dann über das *TransactionTool* gebucht werden.
Automatisierte Kreditkartenzahlungen können über *CyberCash* erfolgen; *CreditCardSSLPaymentClient* kann verwendet werden, wenn dieser Dienst nicht zur Verfügung steht.

4.3.2.4 ECashPaymentClient

Der *ECashPaymentClient* implementiert die Methode *invoiceRequest* für Zahlungen über das elektronische Zahlungsverfahren ECash.
Zunächst wird auch hier die vorher angelegte Transaktion aus der Datenbank geholt. Dann wird getestet, ob die ECash-Umgebung korrekt ist, d.h. ob die ECash-Software und die persönliche ECash-Geldbörse an dem Ort sind, der in der Datei *ecash.props* angegeben wurde.
Im Fehlerfall wird der aufgetretene Fehler in die Transaktion in der Datenbank geschrieben und ein InvoiceResult-Objekt mit Fehlerbeschreibung zurückgegeben.
Im anderen Fall wird ein Systemaufruf gestartet, der die ECash-Software mit den entsprechenden Zahlungsdaten so aufruft, dass die entstehende Zahlungsaufforderung ausgegeben wird. Diese Ausgabe des Prozesses wird nun über einen *BufferedReader* bis zum Ende eingelesen. Durch den Aufruf der ECash-Software wartet der ECash-Server bis zu einem festgelegten *TimeOut* auf die Durchführung der Zahlung, deren Aufforderung soeben gestellt wurde. Deshalb wird nun ein Thread gestartet, der auf das Ende des ECash-Serverprozesses wartet. Mittels des Rückgabewertes dieses Prozesses wird dann festgestellt, ob die Zahlung erfolgreich durchgeführt worden ist.
Die Methode gibt im Attribut *invoiceASCII* des InvoiceResult-Objektes die generierte Zahlungsaufforderung zurück. Wird diese ASCII-Nachricht von einer ECash-Wallet gelesen, so wird dort die entsprechende Zahlungsaufforderung gestellt. Deshalb muss die Nachricht von der Software, die den PaymentHandler nutzt, noch auf geeignete Weise an den Kunden geschickt werden (bspw. per MIME-Message oder eMail).

4.3.2.5 CyberCashClient

Zahlungen über das CyberCoin-Verfahren im CyberCash-System werden durch *CyberCashClient* realisiert.
Auch hier wird zuerst wieder die neu angelegte Transaktion aus der Datenbank gelesen. Dann wird ein Prozess gestartet, der das CGI-Skript *invoice.cgi* aufruft. Dieses CGI-Skript erzeugt - ähnlich wie beim ECash-System - als Ausgabe eine Zahlungsaufforderung, die von einer CyberCash-Geldbörse gelesen werden kann. Diese Zahlungsaufforderung wird vom *CyberCashClient* eingelesen. Im Gegensatz zum ECash-System wartet das Skript jedoch nicht, bis die entsprechende Zahlung erfolgt, sondern terminiert sofort nach Ausgabe der Zahlungsaufforderung. Sollte es beim Erzeugen der Aufforderung zu Fehlern kommen, so wird ein entsprechendes *InvoiceResult*-Objekt zurückgegeben.
Rückgabewert der Methode *invoiceRequest* ist ein *InvoiceResult*-Objekt, in dessen Attribut *invoiceASCII* die Zahlungsaufforderung steht. Diese kann per Mail oder MIME-Nachricht an den Kunden geschickt werden.

4.3.3 Erweiterbarkeit des PaymentHandlers

Um weitere Zahlungsverfahren in den PaymentHandler zu integrieren, muss für jedes Verfahren eine entsprechende, von *PaymentClient* abgeleitete Klasse erstellt werden.
Diese Klasse muss in der Methode *invoiceRequest()* die jeweils zahlungssystem-spezifische Implementierung enthalten. Der entsprechende Datensatz der neuen Transaktion wird bereits vor dem Aufruf der jeweiligen Clients von *PaymentHandler* angelegt. Sowohl bei erfolgreich

durchgeführter Zahlung als auch im Fehlerfall sollte dieser Datensatz - soweit möglich - aktualisiert werden.
Rückgabewert der Methode *invoiceRequest* ist ein *invoiceResult*-Objekt. Darin kann das Ergebnis der Zahlungsaufforderung durch einen Fehlercode, eine Text-Nachricht und - insbesondere bei elektronischen Zahlungsverfahren, bei denen die eigentliche Aufforderung über MIME-Messages verschickt wird - als ASCII-Zahlungsaufforderung zurückgegeben werden.
Weiterhin muss dem neuen Zahlungsverfahren eine ID und ein Name zugewiesen werden. Dies geschieht durch Hinzufügen der entsprechenden Konstanten in der Klasse *Brand*.
Um die Abfrage der unterstützten Zahlungsverfahren mittels *getBrandIds()* zu aktualisieren, muss auch diese Methode der Klasse *PaymentHandler* entsprechend erweitert werden.
Schliesslich muss bei einem Aufruf von *invoiceRequest()* in *PaymentHandler* dieser Aufruf analog zu den bereits integrierten Clients an den neuen PaymentClient weitergeleitet werden. Dies geschieht durch Vergleich der an *invoiceRequest()* übergebenen BrandId mit der ID des neuen Zahlungsverfahrens und anschließender Delegation.

4.4 Benutzungshandbuch

4.4.1 Einrichtung und Betrieb der Payment-Komponente

1. **Verschiedenes**
   Zum Betrieb des PaymentHandlers wird ein Betriebssystem mit installiertem TCP/IP-Protokoll und Java 1.2 von SUN benötigt. Es empfiehlt sich, in Hinblick auf die Verwendung der elektronischen Zahlungsverfahren ECash und CyberCash auf das Betriebssystem UNIX zurückzugreifen. Das sogenannte CashRegister von CyberCash läuft zwar auch unter Windows NT, wobei jedoch hier auf ECash verzichtet werden müsste, da die Geldbörse (Wallet) für den ECash-Händler nur für UNIX erhältlich ist.

   Wenn die elektronischen Zahlungsverfahren unter einem UNIX-Betriebssystem eingesetzt werden sollen, ist darauf zu achten, dass der PaymentHandler, die ECash-Händler-Wallet und das CyberCash-CashRegister unter demselben Nutzer eingerichtet werden. Ansonsten kann der PaymentHandler die elektronischen Zahlungsverfahren nicht verwenden. Dies bedingt auch, dass elektronische Zahlungsverfahren und PaymentHandler auf demselben Host installiert werden müssen.

   Die elektronischen Zahlungsverfahren erfordern im Echtbetrieb ein Bankkonto und eine Registrierung bei dem jeweiligen Kreditinstitut, welches das Zahlungsverfahren unterstützt. Für ECash ist ein Demo-Betrieb möglich, der es erlaubt, elektronischen Zahlungsverkehr mit Spielgeld durchzuführen. CyberCash erlaubt leider keinen Demo-Betrieb, was zur Folge hat, dass von Anfang an mit echtem Geld gearbeitet werden muss. CyberCash empfiehlt hierfür zu Testzwecken nur Zahlungen vom eigenen Konto auf das eigene Konto durchzuführen.

   Damit Zahlungen über elektronische Zahlungsverfahren durchgeführt werden können, muss auf Kundenseite eine entsprechende Wallet installiert sein. Die Kunden-Wallets für ECash und CyberCash sind bislang nur für Windows 9x/NT verfügbar. Auch hier ist noch zu erwähnen, dass die Kunden-Wallets zunächst bei dem jeweiligen Kreditinstitut registriert werden müssen, bevor sie für Zahlungen benutzt werden können (für den Demo-Betrieb von ECash gilt dies nicht).

2. **Installation**
   Sämtliche für den PaymentHandler benötigte Klassen sind dem Java-Archiv `DDS.jar` enthalten. Vor dem Start des PaymentHandlers muss zunächst die Konfigurationsdatei `payment.props` bearbeitet werden. Sie enthält die für den Betrieb des Payment-Servers benötigten Einstellungen:

   `PH_SERVER = <IP-Adresse des Payment-Servers>`
   Gibt die Adresse des Hosts an, auf dem der Payment-Server gestartet wird. RMI benötigt diese Angabe, um zu wissen auf welcher IP-Adresse Anfragen erwartet werden (theoretisch könnten einem Host mehrere IP-Adressen zugeordnet sein).

   `PH_PORT = <Portnummer des Payment-Servers>`
   Der Payment-Server erzeugt sich beim Start eine eigene RMI-Registry. Die RMI-Registry belegt diese Portnummer. Sie muss vorher also frei sein.

   `PH_POLICY = <kompletter Pfad der Policy-Datei>`
   In der Policy-Datei können Rechte für Dateizugriffe gesetzt werden und Ports gegen Verbindungen geschützt werden. Seit Java 1.2 muss diese Datei für RMI verwendet werden.

   `PH_CLASSES = <URL der Server-Klassen>`
   Ort der Payment-Server-Klassen, so dass sich ein Client bei Bedarf Klassen herunterladen kann.

   `ECASH_EXE = <kompletter Pfad der ECash-Händlerbörse>`
   Hier wird der Pfad zu der ausführbaren ECash-Händlerbörse eingetragen.

   `ECASH_HOME = <Pfad zum ECash-Home-Verzeichnis>`
   Das ECash-Home-Verzeichnis beinhaltet ein Unterverzeichnis, in dem die ECash-relevanten Daten gespeichert werden.

   `INVOICE_SCRIPT = <kompletter Pfad des CyberCash invoice.cgi>`
   Bei Zahlungen über CyberCash ruft der PaymentHandler dieses cgi-Skript, um eine Zahlungsaufforderung zu generieren.

   Zum Start des Payment-Servers (PaymentHandler) muessen neben der Datei `DDS.jar` auch die für die Datenbankzugriffe benötigten Treiber im `CLASSPATH` eingebunden werden. Außerdem werden die Pfade zu den Konfigurationsdateien des PaymentHandlers benötigt. Neben der Datei `payment.props` wird noch die Datei `webshop.ini` gebraucht, die noch einige für die Datenbank wichtige Parameter enthält. Sind diese Vorarbeiten gemacht, kann der PaymentHandler wie folgt gestartet werden:

```
java -Dpayment.props=<Pfad zur Datei payment.props>
     -Dwebshop.ini=<Pfad zur Datei webshop.ini>
     webshop.payment.PaymentHandler
```

3. **Hinweise zu Nutzung des RMI-Clients**
   Wie schon in der Schnittstellendokumentation (Kap 4.2.1) beschrieben, nutzen alle Komponenten des DDS-Systems für Zugriffe auf den PaymentHandler die Klasse *RMIClient*. Damit die Nutzung des RMI-Clients reibungslos ablaufen kann, müssen in der Konfigurationsdatei des DDS-Systems (`webshop.ini`) einige Parameter zum PaymentHandler gepflegt werden. Die folgenden Parameter sind ein Auszug aus der Datei `webshop.ini`. Sie sind essentiell für den RMI-Client.

`PH_ShopID = <Identifikationsnummer des DDS>`

Die Kombination aus ShopID und ShopPassword bildet den Login, der für die Authentifizierung beim PaymentHandler benötigt wird. Der RMI-Client benötigt diesen Login-Daten bei den meisten Methodenaufrufen des PaymentHandlers (s. Kap 4.2.1). Die Login-Daten müssen in der Datenbank des PaymentHandlers hinterlegt sein, damit eine einwandfreie Authentifzierung möglich ist (siehe auch **Kapitel 3: Datenbank-Komponente**).

`PH_ShopPassword = <Passwort zur Authentifizierung des DDS>`

siehe oben

`PH_ServerURL = <IP-Adresse des Payment-Servers>`

Über diese Adresse findet der RMI-Client den Host des PaymentHandlers.

`PH_ServerPort = <Portnummer des Payment-Servers>`

Die Portnummer unter der die RMI-Registry des PaymentHandlers läuft.

`PH_PolicyFile = <kompletter Pfad der Policy-Datei>`

In der Policy-Datei können Rechte für Dateizugriffe gesetzt werden und Ports gegen Verbindungen geschützt werden. Seit Java 1.2 muss diese Datei für RMI verwendet werden.

4.4.2 Das Administrations-Tool zur Transaktions-Verwaltung

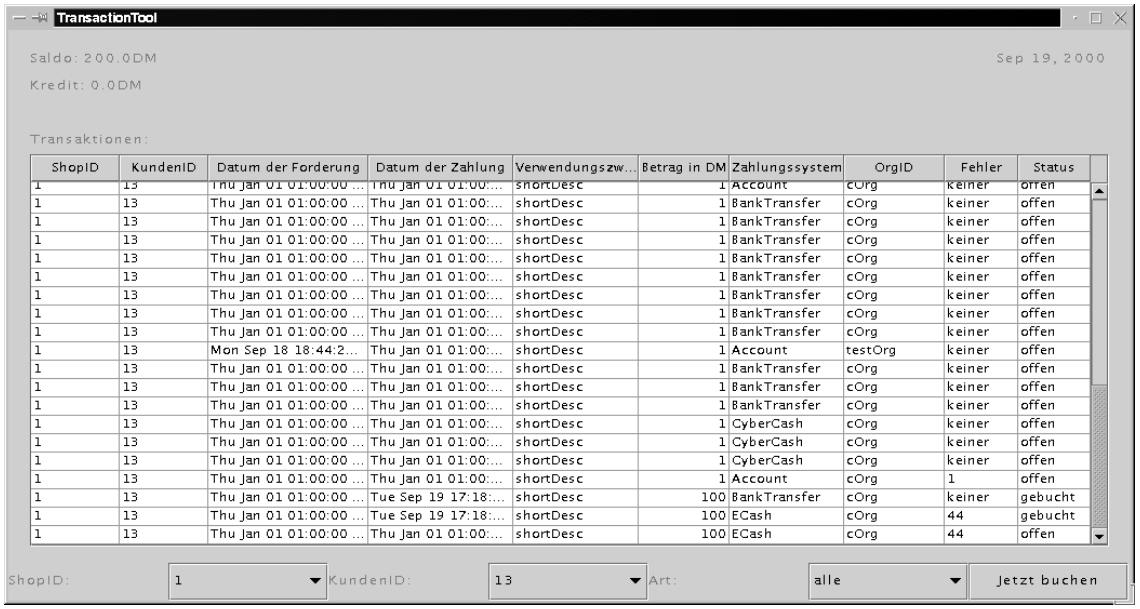

Abbildung 4.2: Screenshot des Administrations-Tools

Zur Verwaltung der durchgeführten Transaktionen wurde ein zusätzliches Administrations-Tool für den PaymentHandler entwickelt. Dieses dient dazu, übersichtlich alle Transaktionen mit allen relevanten Informationen (z.B. Fehler, Buchung, Betrag etc.) anzuzeigen. Weitere Aufgabe ist es, eine geeignete Oberfläche zur Verfügung zu stellen, um gegebenenfalls den Status von Transaktionen manuell zu ändern, falls es bei der automatischen Buchung durch den PaymentHandler zu Fehlern kam. Insbesondere können Zahlungen durch Banküberweisungen durch das Administrations-Tool gebucht werden, da für diese eine automatische Buchung aufgrund der fehlenden Schnittstelle nicht möglich ist.
Die Funktionalität des Administrations-Tools ist durch die einfache Oberflächengestaltung leicht erfassbar; deshalb soll hier nur eine kurze Erklärung der angezeigten Informationen und der Funktionen erfolgen.

Saldo :
Gibt den aktuellen Kontostand des gewählten Kunden an (nur verfügbar, falls Kunde ausgewählt ist).

Kredit :
Gibt des Überziehungskredit an, der dem Kunden gewährt wird (ebenso nur verfügbar, falls Kunde ausgewählt ist) .

ShopId :
Gibt die Id des Shops an, auf den sich die Transaktion bezieht.

KundenId :
Gibt die Id des Kunden an, der die Transaktion initiert hat.

Datum der Zahlungsaufforderung :
Das Datum, an dem die Zahlungsaufforderung gestellt wurde.

Datum der Zahlung :
Das Datum, an dem die Zahlung durchgeführt wurde.

Verwendungszweck :
Der Verwendungszweck, der bei der Zahlungsaufforderung angegeben wurde.

Betrag :
Der Betrag der Transaktion in DEM.

Zahlungssystem :
Das zur Zahlung verwendete Zahlungssystem.

OrgId :
Zusätzliche Informationen über den Kunden, z.B. IP-Adresse o.ä. .

Fehler :
Falls während der Zahlung ein Fehler aufgetreten ist, wird dies hier angezeigt.

Status :
Der Status der Transaktion. Dieser ist entweder 'Zahlungsaufforderung gestellt' oder 'Gebucht'.

Button 'Shop' :
Hier kann der Shop ausgewählt werden, dessen Transaktionen angezeigt werden sollen. 'Alle' zeigt die Transaktionen aller Shops an.

Button 'Kunde' :
Der Button 'Kunde' dient der Auswahl des Kunden, dessen Transaktionen angezeigt werden sollen. Auch hier führt die Auswahl 'Alle' zur Anzeige der Transaktionen aller Kunden.

Button 'Art' :
Durch den Button 'Art' kann die Anzeige der Transaktionen auf einen bestimmten Transaktions-Status beschränkt werden. 'Alle' zeigt alle Transaktionen an; 'Offene' zeigt nur die Transaktionen an, bei denen die Zahlung noch aussteht. 'Gebuchte' zeigt dagegen die Transaktionen an, bei denen die Zahlung erfolgt ist und die bereits gebucht wurden.

Button 'Jetzt buchen' :
Durch diesen Button wird die aktuell gewählte Transaktion gebucht. Dies sollte nur erfolgen, falls die ensprechende Zahlung auch eingegangen ist.

Kapitel 5

Dokumentschutz-Komponente

5.1 Motivation

Um die Interessen der Autoren hinsichtlich Copyright-Verletzungen zu schützen, ist es notwendig, Sicherheitsmaßnahmen zu ergreifen, die das unerlaubte Einsehen von geschützten Dokumenten verhindern. Allerdings gilt es hierbei zu bedenken, dass der User nicht zu viele Sicherheitsbarrieren überwinden muss, bevor er ein Dokument lesen kann. Es sollte also ein Kompromiss zwischen der Sicherheit der Dokumente und dem Bedienungskomfort gefunden werden, der sowohl die Autoren der Dokumente als auch den User zufriedenstellt. In den nächsten Kapiteln wird dargestellt, welche Schutzmechanismen existieren und welche davon für das DDS-System als sinnvoll angesehen wurden.

5.2 Das Lizenz-Konzept

Das generelle Lizenz-Konzept sieht vor, dass der User eine zeitlich befristete Lizenz beim Webshop für ein Dokument erwirbt und dann solange in dem Dokument navigieren kann, wie die Lizenz gültig ist. Der User kann zwar über die Lizenzdauer hinaus die geladenen Dokument-Teile lesen, aber er kann nicht über die Lizenzdauer hinaus über die im Dokument enthaltenen Links weiter in dem Dokument navigieren, was ein Weiterlesen des Dokuments verhindert.
Bei der Umsetzung des Lizenz-Verfahrens für den Webshop wird das Überprüfen der Gültigkeit der Lizenz über Session-ID's realisiert. Das bedeutet, der User bekommt beim Kauf der Lizenz einen Link zu dem gewünschten Dokument. In diesem Link ist eine eindeutige Zeichenkette (Session-Id) enthalten, welche bei Anfragen an den Webserver ausgelesen wird und über die in Zusammenhang mit dem Link auf das Dokument die User-Daten und die Gültigkeit der Lizenz ermittelt werden. Anhand dieser Daten wird entschieden, ob der User weiter in dem Dokument navigieren darf, oder nicht. Darf der User weiternavigieren, so wird ihm mit der angeforderten Seite des Dokuments eine neue Session-ID mitgeliefert, die automatisch in allen Links des Dokuments enthalten sind (dies wird im Entwurf genauer erläutert). Mit dieser neuen Session-Id kann er eine bestimmte Zeit navigieren, bis ihm entweder eine neue Session-ID generiert wird, oder er eine neue Lizenz erwerben bzw. das Navigieren in dem Dokument einstellen muss. Diese session-ID-Vergabe wird über die komplette Lizenzdauer durchgeführt.
Dieses Lizenz-Konzept verhindert jedoch nicht, dass der User nach dem Kauf einer kurzen Lizenzzeit das gesamte Dokument automatisch vom Server herunterlädt, was nicht im Sinne des

Konzepts ist. Der User soll regelmäßig neue Lizenzen erwerben, sofern er weiter in dem Dokument lesen möchte. Wenn er die Gelegenheit hat, das komplette Dokument herunterzuladen, braucht er danach keine neuen Lizenzen, weil er das Dokument dann nämlich kostenlos ohne eine Internetverbindung lesen kann. Deswegen müssen weitere Sicherheitstechniken betrachtet werden, die im nächsten Unterabschnitt aufgezählt und erläutert werden.

5.2.1 Eine Aufzählung von möglichen Sicherheitstechniken , die auf das Lizenz-Konzept aufsetzen bzw. es ersetzen

1. PDF-Verschlüsselung

   Bei PDF-Dateien ist ein Bereich der Datei für Algorithmen und die Speicherung von Daten dieser Algorithmen vorgesehen. Dies würde die Speicherung des Datums des ersten Aufrufs ermöglichen, welches verschiedene Möglichkeiten zum Schützen des Dokuments bereit stellt. Dazu gehören u.a. das einmalige Anzeigen des Dokuments, wonach das Dokument nicht mehr geöffnet werden kann, oder das Öffnen des Dokuments nur auf eine bestimmte Zeit zu beschränken.
   Mit ähnlichen Verfahren kann auch das Ausdrucken des Dokuments verhindert werden.

2. Versteckte Links in HTML-Dateien

   Die Idee dieser versteckten Links ist, den User beim illegalen Herunterladen von Dokumenten extrem viel unnütze Daten mitzuschicken, die den Download erschweren, bzw. unendlich machen. Dabei werden versteckte Links auf freie Dokumente in die HTML-Dateien eingebaut, was dazu führt, dass diese Dokumente mit heruntergeladen werden, wenn das eigentlich gewünschte Dokument automatisch heruntergeladen wird. Auf eine Zeile Text würden viele Megabytes an überflüssigen Daten mitgeladen werden. Dies passiert aber nur, wenn automatisch das gesamte Dokument per rekursivem Download geladen wird, z.B. mit dem Programm wget.
   Dieses Verfahren ist auf unterschiedliche Weise anwendbar. Man könnte z.B. aus jedem Satzendezeichen (Punkt, Ausrufezeichen, Fragezeichen, etc.) Einen Link machen, der auf ein größeres freies Dokument verweist, z.B. Selfhtml oder die Java-API. Wenn diese Verweise Servlet-Aufrufe beinhalten würden, wäre es sogar vorstellbar, immer wieder das gleiche Dokument mit einem anderen Verzeichnisnamen zu verschicken.

3. Ein eigenes Client-Server-System

   Der Grund, weswegen mit Programmen wie wget komplette Dokumente vom Webserver in kürzester Zeit heruntergeladen werden können ist, dass es sich um Standard-Internet-Dateien (HTML, PDF, etc) handelt, die von einem Standard-Internet-Server (Apache) bereitgestellt werden. Aufgrund dieser Standards gibt es viele Möglichkeiten, das Laden von Dokumenten durchzuführen.
   Über ein eigenes Client-Server-System, welches eine proprietäre Schnittstelle besitzt, die nur bestimmten Software-Entwicklern zugängig ist, kann man die Nutzung solcher Programme unterbinden. Niemand wäre in der Lage, an Dokumente zu gelangen, ohne das spezielle Client-Programm zu benutzen, weil alle anderen Client-Programme aufgrund der proprietären Schnittstelle nicht in der Lage sind, mit dem Server, auf dem die Dokumente liegen, zu kommunizieren.

4. Die Dokumente selbst verschlüsseln

   Eine Alternative zur Verhinderung des Herunterladens der Dokumente besteht in der Verschlüsselung der Dokumente. Würde jedes Dokument verschlüsselt, so wäre es nur mit einem speziellen Client-Programm bzw. einem PlugIn auf Client-Seite möglich, die Dokumente zu dekodieren und zu lesen. Das bedeutet, dass zwar jeder Dokumente komplett in kürzester Zeit vom Server herunterladen , sie aber nicht lesen kann, sofern er sie nicht ordnungsgemäß (d.h. mit einer gültigen Lizenz über einen speziellen Client bzw. einen Client mit speziellem PlugIn) heruntergeladen hat.

5. Die Dokumente verkaufen

   Anstelle die Dokumente nur zeitlich befristet zum Laden und Lesen zur Verfügung zu stellen, werden sie komplett zu einem entsprechenden Preis zum Verkauf angeboten. Das würde eine Verschlüsselung oder Erschwerung des kompletten Herunterladens der Dokumente überflüssig machen.

6. Umbenennung der Dateinamen beim Herunterladen
   Wenn man die Namen der Dateien, die über den Klick auf einen Link angefordert werden, umbenennt, erhält der User das gesamte Dokument, aber er kann nicht auf der eigenen Festplatte zu Hause darauf navigieren. Das liegt daran, dass die Namen in den Links nicht mehr mit den Namen auf der Festplatte korrespondieren. Diesen Zustand wieder herzustellen, würde die Umbennung der Dateien auf der Festplatte voraussetzen, d.h. der User muss zu jedem Link die entsprechende Datei finden und sie entsprechend dem Link zurück umbenennen. Dies ist allerdings bei Dokumenten, die aus mehreren hundert Dateien bestehen, sehr aufwendig.

5.3 Analyse und Problembetrachtung der unterschiedlichen Sicherheitstechniken

Es gibt mehrere Probleme, die beim weiteren Vorgehen betrachtet werden müssen und die Möglichkeiten der weitergehenden Sicherheitstechniken einschränken :

1. Der User darf durch die Technik nicht beeinträchtigt werden

   Der User soll möglichst nicht merken, welche Techniken im Hintergrund arbeiten. Er zahlt Geld dafür, dass er Dokumente lesen darf. Mit Installationsmaßnahmen sollte er nicht belästigt werden, weil ihn das im Zweifelsfall davon abhält, den Webshop zu benutzen.

2. Die interne Struktur und der Inhalt der Dokumente soll nicht verändert werden

   Die interne Veränderung der Dokumente ist nicht generell möglich. Das bezieht sich nicht nur auf den Inhalt der Texte und Grafiken, diese Inhalte sollen sowieso nicht verändert werden (Urheberrecht). Es geht dabei auch um den Quelltext, der um die Text-Daten herum existiert. Es wäre z.B. problematisch, wenn Dokumente aufgrund

von Sicherheitstechniken, die die interne Struktur des Dokuments verändern, auf einigen Clients nicht mehr angezeigt werden könnten, bzw. fehlerhaft angezeigt werden, was dann dem Webshop-Betreiber von den Firmen, die Dokumente über den Webshop einstellen wollen, in Rechnung gestellt werden kann.
Deswegen darf der Inhalt der Dokumente nicht verändert werden!

3. PDF-Dokumente alleine sind nicht ausreichend

   Das Webshop-Prinzip auf PDF-Dokumente zu beschränken, würde die Lizenz-Vergabe und das Session-ID-Verfahren überflüssig machen. Da das Ziel der Projektgruppe ein Webshop für viele unterschiedliche Dokument-Arten ist, wäre die Verschlüsselung von PDF-Dateien nur als Erweiterung des Webshops in Betracht zu ziehen.

Unter diesen Einschränkungen werden jetzt die unterschiedlichen weitergehenden Sicherheitstechniken analysiert :

PDF-Dokumente verschlüsseln	Dies kann nur als Erweiterung des Webshops in Betracht gezogen werden, jedoch nicht als generelle Maßnahme.
Versteckte Links in HTML-Dateien	Dieses Verfahren würde die interne Struktur der HTML-Dateien verändern, weswegen dieses Verfahren ausscheidet.
Ein eigenes Client-Server-System	Der User müßte sich das Client-Programm von einer Webseite herunterladen und es installieren. Damit würde man den User belästigen und zusätzlich die Gefahr eingehen, dass durch die Installation oder sein eigenes Handeln die Funktionalität seines Computer in Gefahr ist, weswegen der User im Zweifelsfall Abstand von der Nutzung des Webshops nehmen wird.
Die Dokumente selbst verschlüsseln	Der User müßte sich das Client-Programm bzw. das PlugIn für seinen Client, welches die Dokumente entschlüsselt aus dem Internet laden und installieren. Das belästigt den User, hat die gleichen Nachteile wie das Client-Server-System und setzt möglicherweise auch noch einen entsprechenden Rechner beim User voraus, der in der Lage ist, die Dokumente in Echtzeit zu dekodieren, um den User nicht unnötig warten zu lassen, bis er das Dokument lesen kann.
Die Dokumente verkaufen	Das ist die einfachste Methode, jedoch stellt sie keine Alternative dar, weil genau dieses Verfahren im Gegensatz zu den Webshop-Prinzipien steht. Deswegen scheidet dieses Verfahren aus.
Umbenennung der Dateinamen	Dieses Verfahren erschien der Projektgruppe am interessantesten, weil es keine der oben genannten Einschränkungen unterliegt und gleichzeitig einen gewissen Schutz bietet. Warum dieses Verfahren jedoch nicht zum Einsatz kam, obwohl viel Zeit in die Entwicklung investiert wurde, wird im Abschnitt 5.8 erläutert.

5.4 Entwurf

5.4.1 Anfrage über Socket-Verbindung

Das C-Modul sendet die Session-ID und die URI per Socket-Verbindung an das Security-Servlet. Das Security-Servlet stellt eine Datenbankanfrage mit der Session-ID und der URI und bekommt Informationen, die im weiteren Verlauf ausgewertet werden.

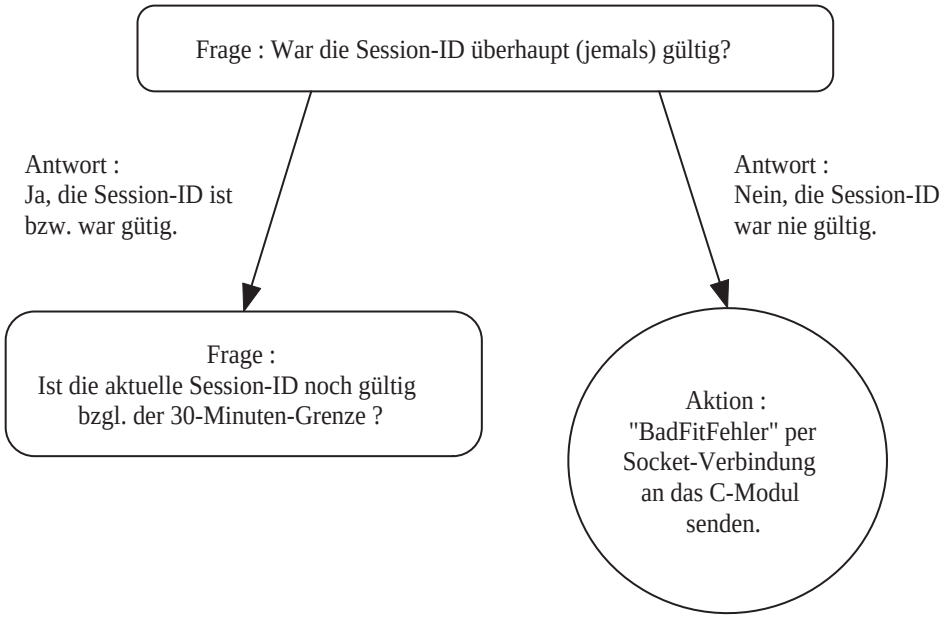

Abbildung 5.1: Ist die Session-ID jemals gültig gewesen?

Es wird gefragt, ob die Session-ID jemals gültig gewesen ist, das bedeutet, ob die Session-ID bzgl. der gesendeten URI in der Datenbank verwendet worden ist. Diese Frage ist nicht davon abhängig, ob sie immer noch gültig ist, sondern ob es sich hier um einen validen Link bzw. Bookmark auf die angeforderte Seite handelt, oder ob jemand versucht, mit einer ausgedachten Session-ID an das Dokument im Webshop zu gelangen, ohne Lizenzgebühren zu bezahlen.
Falls es sich um eine gültige Session-ID handelt, muss ermittelt werden, ob die Session-ID immer noch gültig ist. Die Gültigkeit einer Session-ID ist zeitlich begrenzt, damit sich nicht mehrere User eine Lizenz teilen können. Sollten mehrere User dieses doch versuchen, so wird nach einer bestimmten Zeit (in unserem Beispiel nach 30 Minuten) eine neue Session-ID generiert, die dann nur einem User zur Verfügung steht, es sei denn, er stellt den Link mit der neuen Session-ID wieder den anderen Usern zur Verfügung. Damit ist jedoch ausgeschlossen, dass jemand einen gültigen Link im Internet zur Verfügung stellt, weil er dadurch das Risiko eingeht, für eine Lizenz bezahlt zu haben, die er selber nicht komplett ausnutzen kann.
Falls die Session-ID nicht in der Datenbank enthalten ist, war sie nie einem Dokument zugeordnet und damit nie gültig. In diesem Fall wird eine BadFitException vom Datenbankwrapper geworfen. Um dem User mitzuteilen, dass er mit dieser Session-ID nicht an sein lizensiertes Dokument kommen kann, sendet das Security-Servlet den String „BadFitFehler“ per Socket-Verbindung an das C-Modul, welches dadurch weiss, welche Ausgabe auf dem Browser des Users durchgeführt werden muss.

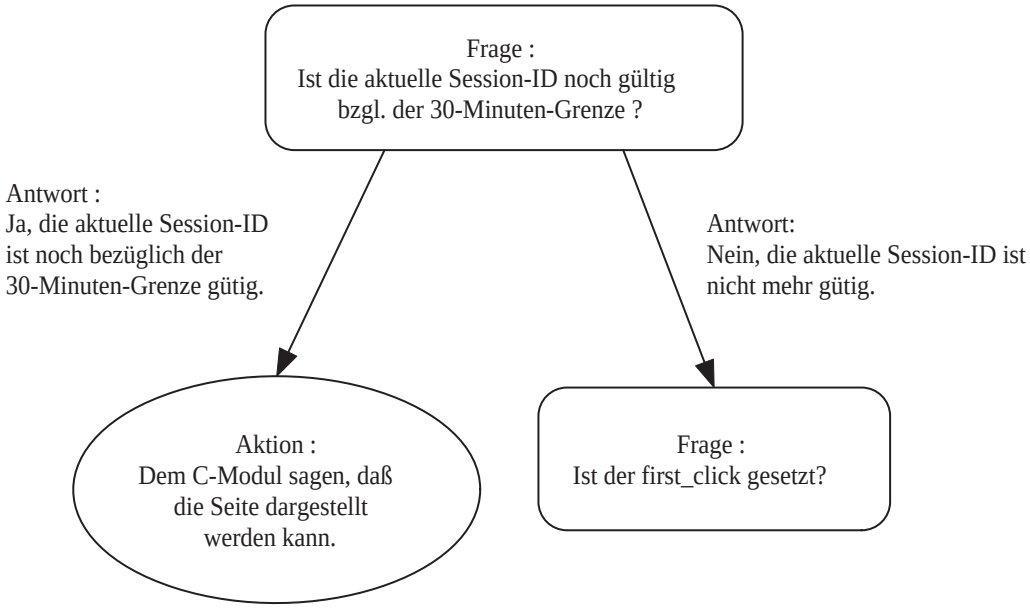

Abbildung 5.2: Ist die Session-ID noch gültig bzgl. der 30-Minuten-Grenze?

Wie schon oben beschrieben, ist die Gültigkeit einer Session-ID zeitlich begrenzt, um zu verhindern, dass Webshop-Lizenzen über das Internet von mehreren Usern gleichzeitig genutzt werden. In diesem Beispiel ist die Grenze auf 30 Minuten gesetzt.
Ist die aktuelle Session-ID noch bezüglich der 30-Minuten-Grenze gültig, müssen die Daten des Users nicht weiter überprüft werden. Deshalb sendet das Security-Servlet den String „ok“ per Socket-Verbindung an das C-Modul. Das C-Modul reagiert auf diesen String, indem es die angeforderte Seite auf dem Browser des Users darstellt.
Ist die aktuelle Session-ID nicht mehr bezüglich der 30-Minuten-Grenze gültig, so kann das mehrere Gründe haben. In jeden Link auf Dokumente des Webshops wird eine Session-ID eingebaut, damit das C-Modul weiss, dass diese Links überprüft werden müssen. Jedoch soll die Zeit, die ein User über eine Lizenz zum Lesen der Dokumente gekauft hat, erst heruntergezählt werden, sobald er das erste Mal auf das Dokument zugreift. Deshalb wird dem User im Webshop ein Link auf das lizensierte Dokument zur Verfügung gestellt, welcher eine in der Datenbank eingetragene Session-ID enthält, die jedoch schon abgelaufen ist. Erst wenn der first_click, also der erste Zugriff auf ein lizensiertes Dokument, in der Datenbank eingetragen ist, wird die eingekaufte Zeit heruntergezählt. Deswegen wird auch erst gefragt, ob der first_click gesetzt worden ist, bevor ermittelt wird, ob die Lizenz überhaupt noch gültig ist.

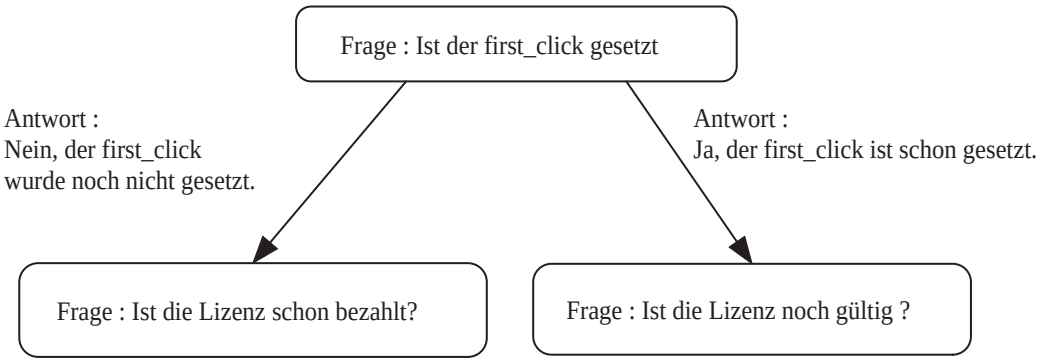
Frage : Ist der first_click gesetzt
Antwort :
Nein, der first_click
wurde noch nicht gesetzt.
Antwort :
Ja, der first_click ist schon gesetzt.
Frage : Ist die Lizenz schon bezahlt?
Frage : Ist die Lizenz noch gültig ?

Abbildung 5.3: Ist der first_click gesetzt?

Ist der first_click gesetzt, d.h. ist die Lizenz überhaupt schon genutzt worden, oder handelt es sich hier um den ersten Zugriff auf ein lizensiertes Dokument?
Falls der first_click noch nicht gesetzt worden ist, handelt es sich hier um den ersten Aufruf des lizensierten Dokuments oder die Lizenz ist noch nicht bezahlt worden, weswegen noch kein Zugriff auf das Dokument erlaubt wurde. Deswegen wird hier gefragt, ob die Lizenz schon bezahlt worden ist. Dieser Fall ist nicht selten, weil einige Zahlungen über das Intenet mit einer Zeitverzögerung von den Banken bestätigt werden, weshalb nicht bei jedem Zahlungsverfahren sofort ein Zugriff auf die gewünschten Seiten gewährt werden kann.
Ist der first_click jedoch schon gesetzt worden, so kann das zwei mögliche Gründe haben, nämlich dass entweder nur die Session-ID abgelaufen ist, oder dass die Lizenz abgelaufen ist. Deshalb muss hier gefragt werden, ob die Lizenz noch gültig ist, damit eine neue Session-ID vergeben werden darf.

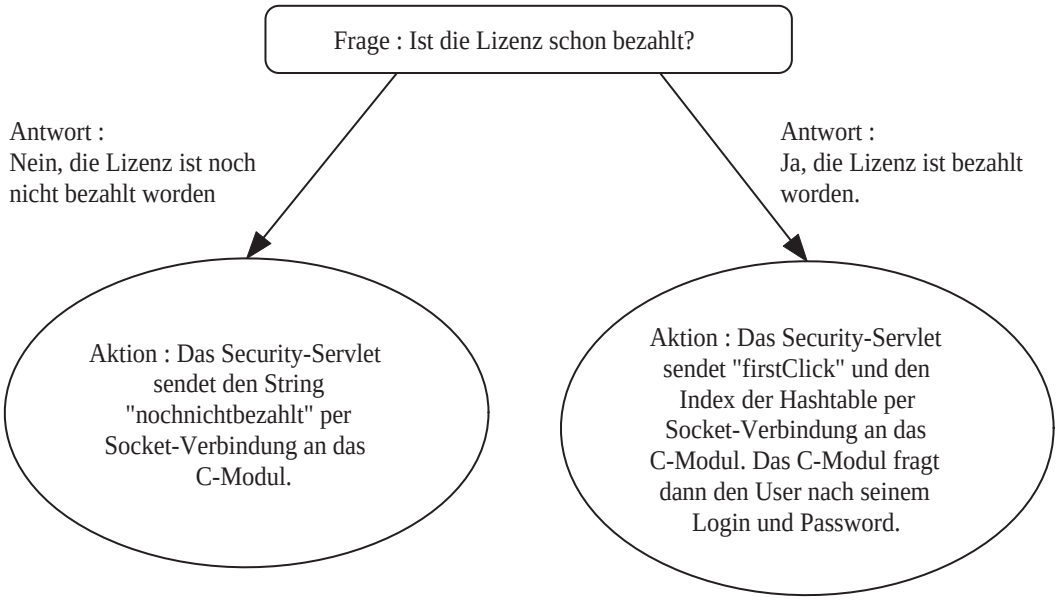

Abbildung 5.4: Ist die Lizenz schon bezahlt?

Hier stellt sich die Frage, ob die Lizenz schon bezahlt ist. Es kann sein, dass zwar schon versucht wurde, auf die Seiten zuzugreifen, die Lizenz jedoch noch nicht freigeschaltet wurde, weil Sie noch nicht bezahlt worden ist bzw. die Information über die erfolgte Zahlung noch nicht eingegangen ist.
Ist die Lizenz noch nicht bezahlt worden bzw. die Information über die Zahlung noch nicht eingegangen, so sendet das Security-Servlet den String „nochnichtbezahlt“ per Socket-Verbindung an das C-Modul, welches eine entsprechende Ausgabe auf dem Browser des Users macht.
Sollte sie doch schon bezahlt worden sein, so handelt es sich hier um den ersten Aufruf der Seiten nach dem Eingang der Zahlung, weswegen der first_click gesetzt werden muss. Dafür muss der User sich jedoch authentifizieren, weswegen das Security-Servlet den String „firstClick“ und den Index der Hashtable, in der die Daten der Lizenz gespeichert sind, per Socket-Verbindung an das C-Modul sendet. Der String „firstClick“ ist die Information für das C-Modul, dass es sich um einen first_click-Request handelt und es den User nach seinem Login und Passwort fragen muss. Danach ruft das C-Modul das Security-Servlet mit dem String „firstClick“, dem Login, Passwort und dem Index in der Hashtable auf.

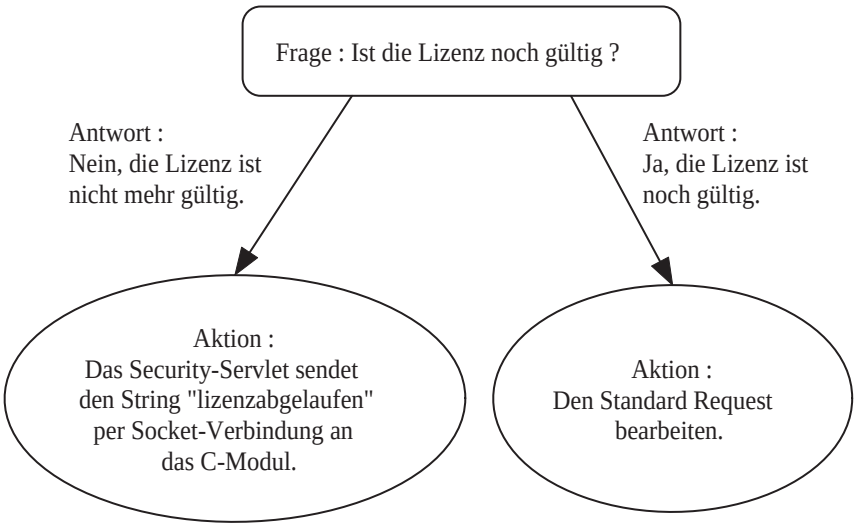

Abbildung 5.5: Ist die Lizenz noch gültig?

Wurde der first_click schon gesetzt, die Session-ID jedoch als abgelaufen angesehen, so muss überprüft werden, ob die Lizenz noch gültig ist, denn nur in diesem Fall darf eine neue Session-ID generiert werden.
Ist die Lizenz nicht mehr gültig, so muss der User eine neue Lizenz kaufen, wenn er weiterhin auf das Dokument im Webshop zugreifen möchte. Damit der User dies erfährt, schickt das Security-Servlet den String „lizenzabgelaufen“ an das C-Modul, welches den User mit einer Ausgabe auf seinem Browser darüber informiert.
Ist die Lizenz noch gültig, dann muss der User sich authentifizieren, damit er eine neue Session-ID generiert bekommt, mit der er weiter in dem Dokument navigieren kann. Dafür sendet das Security-Servlet den String „standard“ und den Index der Hashtable, unter dem die Daten der Lizenz gespeichert sind, an das C-Modul. Mit dem String „standard“ erfährt das C-Modul, dass es sich um einen Standard-Request handelt. Das C-Modul fragt dann den User nach seinem Login und Passwort und ruft das Security-Servlet mit dem String „standard“, dem Login, Passwort und dem Index für die Hashtable auf.

5.4.2 Der first_click-Request

Das C-Modul gibt dem User auf seinem Browser eine Seite aus, in der er nach seinem Login und Passwort gefragt wird. Der String „firstClick“, das Login, Passwort und der zum User gehörende Index in der Hashtable des Security-Servlets werden an das Security-Servlet gesendet. Das Security-Servlet weiss aufgrund des Strings „firstClick“, dass es sich um einen first_click-Request handelt und muss dann entscheiden, ob der User weiter in dem Dokument navigieren darf.

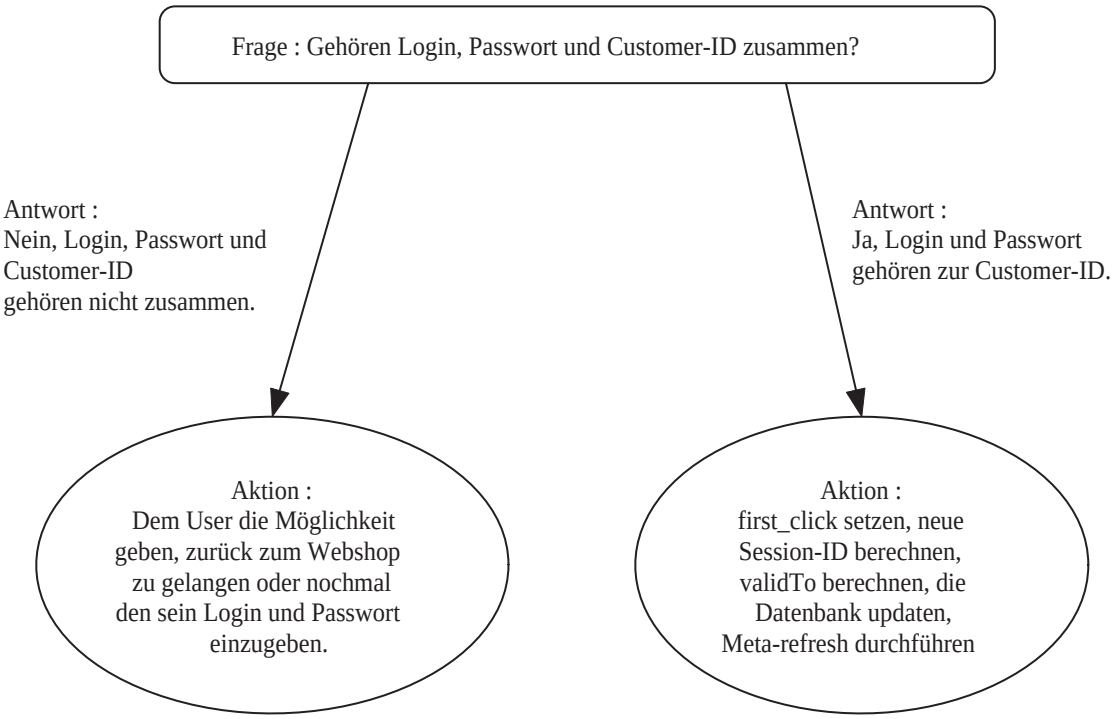

Abbildung 5.6: Der first_click-Request

Gehören Login, Passwort und Customer-ID nicht zusammen, so wird vom Security-Servlet eine Seite ausgegeben, in der dem User die Möglichkeit gegeben wird, zurück zum Webshop zu kommen oder nochmal den first_click-Request zu durchlaufen, um sein Login und Passwort einzugeben, falls er meint, sich vertippt zu haben.
Gehören jedoch Login und Passwort zur Customer-ID, so setzt das Security-Servlet den first_click, berechnet eine neue Session-ID und ein validTo unter der Berücksichtigung der duration, schreibt beides in die Datenbank und führt ein Redirect per Meta-Refresh auf die angeforderte Seite durchgeführt. Danach kann der User wie gewohnt weiter in dem Dokument navigieren.

5.4.3 Der Standard-Request

Das C-Modul gibt dem User auf seinem Browser eine Seite aus, in der er nach seinem Login und Passwort gefragt wird. Der String „standard“, das Login, Passwort und der zum User gehörende Index in der Hashtable des Security-Servlets werden an das Security-Servlet gesendet. Durch den String „standard“ weiss das Security-Servlet, dass es sich um einen Standard-Request handelt und muss dann entscheiden, ob der User weiter in dem Dokument navigieren darf.

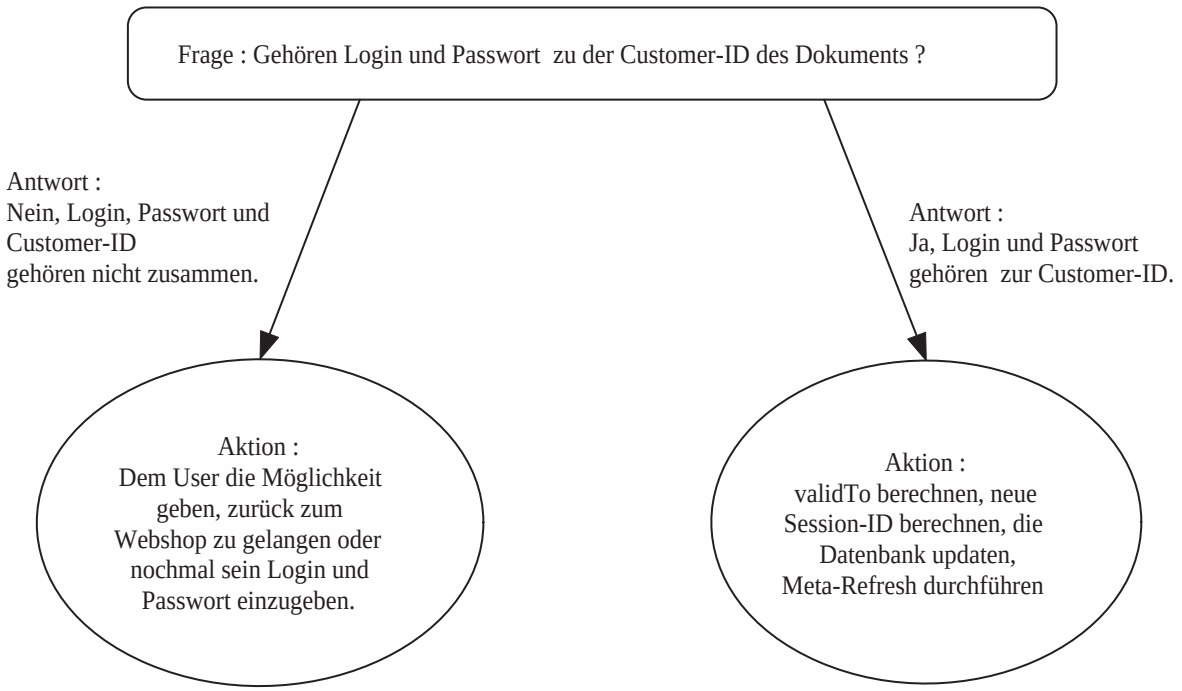

Abbildung 5.7: Der Standard-Request

Gehören Login, Passwort und Customer-ID nicht zusammen, so wird vom Security-Servlet eine Seite ausgegeben, in der dem User die Möglichkeit gegeben wird, zurück zum Webshop zu kommen oder nochmal den Standard-Request zu durchlaufen, um sein Login und Passwort einzugeben, falls er meint, sich vertippt zu haben.
Gehören Login, Passwort und Customer-ID zusammen, so berechnet das Security-Servlet das neue validTo unter der Berücksichtigung der duration, generiert eine neue Session-ID, und schreibt validTo und Session-ID in die Datenbank. Danach wird ein Redirect per Meta-Refresh auf die angeforderte Seite durchgeführt. Der User kann danach weiter in dem Dokument navigieren.

5.5 Realisierung der gewählten Sicherheitstechniken

Als Basis für die Umsetzung der im Entwurf angegebenen Sicherheitstechniken, dient die Webserver-Distribution von Apache. Zum einen ist sie kostenlos, und des weiteren erlaubt sie das einfache Einbinden von C-Module in den Kernel des Servers. Außerdem unterstützt der Apache-Server die Servlet-Technologie, die für den Einsatz von Servlets zwingend erforderlich ist. Somit

unterteilt sich die Implementation der Sicherheitstechniken in die Erstellung des Server-Plugins, das die Anfragen an den Server abfängt, und des Security-Servlets, welches die Anfragen bearbeitet und an den Server zurückleitet.

5.5.1 Implementation des Server-Plugins

Um die Funktionsweise des Server-Plugins zu verstehen, ist es zunächst einmal notwendig zu erklären, wie der Apache-Server eine Anfrage behandelt. Der Server unterteilt die Anfragebehandlung in mehrere Schritte:

1. Übersetzung der URI in einen Dateinamen
2. Überprüfung, ob der User der ist, der er behauptet zu sein
3. Überprüfung, ob der User berechtigt ist, die Seite zu sehen
4. Den MIME-Typ des angefragten Objekts bestimmen
5. Eine Antwort an den Client zurückschicken
6. Die Anfrage protokollieren

In jeder Phase der Anfragebehandlung werden nun die im Apache integrierten Module daraufhin überprüft, ob sie für die aktuelle Phase eine individuelle Behandlungsroutine aufweisen. Ist dies der Fall, so wird diese ausgeführt. Es können dann folgende Aktionen ausgelöst werden:

- Die Anfrage wird bearbeitet und ein `OK` wird an den Server gesendet.
- Die Anfrage wird nicht bearbeitet und die Konstante `DECLINE` wird an den Server zurückgeliefert. Der Server wird dann die Routine des nächsten Moduls aufrufen.
- Eine Fehlermeldung mit einem HTTP-Fehlercode wird an den Client zurückgesendet.

Liefert eine Routine eines Moduls ein `OK` an den Server zurück, so wird die Anfrage als abgearbeitet betrachtet und alle anderen Module werden ignoriert. Die Reihenfolge, in der die Module in den Server-Kernel integriert werden, spielt also eine entscheidende Rolle. Bei den Phasen der Anfragebehandlung ist die Übersetzung der URI in einen Dateinamen für den Dokumentenschutz von besonderer Bedeutung. Die Abkürzung URI steht für „Uniform Resource Identifiers“. Es handelt sich dabei um eine Zeichenkette, die das Identifizieren von Dokumenten im Internet ermöglicht. Bei einer Anfrage an den Webserver wird jedesmal diese URI in einen Dateinamen umgewandelt, um das entsprechende Dokument auf dem Massenspeicher des Webservers zu finden.
An dieser Stelle setzt das Server-Plugin an. Zunächst einmal wird überprüft, ob das angeforderte Dokument in einem geschützten Verzeichnisbereich des Servers liegt, oder ob der MIME-Typ des angeforderten Dokuments generell ungeschützt ist. Wenn das Dokument ungeschützt ist, dann bricht das C-Modul mit einem `DECLINE` ab, und überläßt die weitere Abarbeitung der Anfrage dem Server. Ist das Dokument jedoch geschützt, so wird versucht aus der URI eine Session-ID zu extrahieren. Diese Session-ID ist eine Zeichenkette, die das Login des Users und seine erworbene

Lizenz beinhaltet, mehr dazu siehe Abschnitt 5.2. Diese Session-ID steht immer am Anfang der URI. Konnte eine gültige Session-ID gefunden werden, so wird diese zusammen mit der URI ohne Session-ID an das Security-Servlet geschickt. Wie die Kommunikation genau funktioniert, wird später im Abschnitt „Kommunikation zwischen Server-Plugin und Security-Servlet“ beschrieben. Die Zeichenkette, die vom Servlet zurückgeschickt wird, speichert das Server-Plugin in einer internen Datenstruktur ab. Damit ist die erste Phase der Anfragebearbeitung abgeschlossen.
Der nächste interessante Schritt ist das Zurücksenden einer Antwort an den Client. Hier wird nun in der internen Datenstruktur des Server-Plugins die Antwort des Servlets überprüft. Ist die Antwort `OK` , so wird das angeforderte Dokument angezeigt, wobei wichtig ist, dass die Session-ID wieder in der URI steht. Ohne die Session-ID wäre nämlich das weitere Navigieren in einem geschützten Dokument nicht möglich. Wenn jedoch ein Fehler aufgetreten ist, so sendet der Server eine entsprechende Antwort an den Client zurück. Diese Antwort kann einfach nur aus einem Fehlertext, aber auch aus einer Aufforderung zum Einloggen bestehen, falls die Session-ID abgelaufen ist. Die Bearbeitung des Login-Formulars übernimmt dann wieder das Servlet.

5.5.2 Implementation des Security-Servlets

Das Security-Servlet hat die Aufgabe, zu überprüfen, ob ein bestimmter User Zugriff auf ein bestimmtes Dokument hat oder nicht. Dies geschieht mittels Anfragen an die Datenbank via die Datenbankschnittstelle. Sendet der Server nun eine URI und eine Session-ID zur Überprüfung an das Servlet, so fordert es mit diesen beiden Parametern ein LicenceSessionInfo-Objekt von der Datenbankschnittstelle an. Dieses Objekt besitzt folgende Methoden:

- freeURL(): Gibt an, ob es sich um ein ungeschütztes Dokument handelt.
- isSessionKeyValid(): Liefert `true` zurück, wenn der Session-Key noch gültig ist.
- isFirstClickSet(): Hat der User das Dokument angefangen zu lesen, so liefert diese Methode `true` zurück.
- isLicenceValid(): Gibt an, ob die Lizenz noch gültig ist.

Wenn der User keinen Zugriff auf das Dokument hat, dann liefert die Datenbankschnittstelle eine Exception zurück und das Servlet sendet dem Server eine entsprechende Nachricht. Wurde jedoch keine Exception geworfen, so kann anhand des LicenceSessionInfo-Objektes der Zustand der Lizenz überprüft werden. Wenn beispielsweise die Lizenz noch gültig ist, die Session-ID jedoch nicht, dann wird vom Servlet ein Login-Formular zum Client gesendet, um eine neue Session-ID zu erstellen. Vorher müssen jedoch die von der Datenbankschnittstelle angeforderten Daten gesichert werden. Dies geschieht über einen Hashtable, in dem sowohl die URI als auch das LicenceSessionInfo-Objekt unter einer bestimmten Nummer abgelegt werden. Diese Nummer wird dem Formular als Parameter mitgegeben, um die Daten später wieder auslesen zu können. Denn das Servlet übernimmt auch die Abarbeitung des Login-Formulars. Hierbei wird der User und sein Passwort über die Datenbankschnittstelle überprüft. War die Eingabe korrekt, so wird mit dem SessionIDGenerator eine neue Session-ID erzeugt. Diese Session-ID setzt sich aus den Buchstaben „SID“, dem Login des Users und der verstrichenen Zeit seit dem 1.1.1970 in Millisekunden zusammen. Mit Hilfe der neuen Session-ID und der gespeicherten URI aus dem

Hashtable wird nun ein Metarefresh an den Client gesandt, wodurch das gewünschte Dokument dargestellt wird.
Versucht der User zum ersten Mal ein lizensiertes Dokument einzusehen, so wird ebenfalls ein Login-Formular an den Client gesandt. Diesmal wird bei korrekter Eingabe jedoch vor dem Metarefresh überprüft, ob die Lizenz schon bezahlt ist. Wenn dies der Fall ist, so wird der „first-Click“ der Lizenz gesetzt. Damit lässt sich später berechnen, ob die Lizenz noch gültig ist oder nicht.

5.5.3 Kommunikation zwischen Server-Plugin und Security-Servlet

Die Kommunikation zwischen den beiden Komponenten läuft sowohl über eine Socket-Verbindung, als auch über direkte Aufrufe des Servlets durch HTML-Formulare ab. Zur Realisierung der Socket-Kommunikation, startet das Servlet beim Initialisieren einen Socket-Server als eigenständigen Thread. Das Servlet lauscht also ständig auf Anfragen des Servers. Öffnet der Server nun einen Socket-Kanal, so startet das Servlet einen weiteren Thread, um die Anfrage zu bearbeiten. Daher kann das Servlet theoretisch mehrere Anfragen gleichzeitig bearbeiten. Bei dieser Art der Kommunikation zwischen zwei Modulen werden lediglich Zeichenketten über den Datenkanal gesendet. Der Server schickt eine Anfrage und das Servlet sendet eine Antwort zurück. Danach wird die Verbindung abgebrochen und der Thread beendet. Es wird also für jede Anfrage eine neue Verbindung zwischen Server und Servlet aufgebaut.
Bei der Kommunikation über HTML-Formulare wird direkt die service-Methode des Servlets mit den im Formular angegebenen Parametern aufgerufen. Dies geschieht, wenn der Server eine Passwortabfrage an den Client sendet. War das eingegebene Passwort jedoch nicht korrekt, so sendet das Servlet dieselbe Passwortabfrage nocheinmal an den Client. Somit ruft sich das Servlet selbst über ein HTML-Formular auf. Diese Methode der Kommunikation ist allerdings einseitig, da nur das Servlet über die service-Methode verfügt, und der Server keine HTML-Formulare abarbeiten kann.

5.6 Schnittstellenbeschreibung

Wie schon im Abschnitt „Realisierung“ beschrieben, verfügt die Dokumentenschutz–Komponente über zwei Schnittstellen. Zum Einen kann das Security-Servlet direkt über ein HTML-Formular aufgerufen werden. Dabei werden die Parameter in der Methode `service` ausgewertet. Dies geschieht allerdings nur, wenn bei einer Passwortabfrage ein falsches Kennwort eingegeben wurde und sich das Servlet nocheinmal selbst aufruft, um die Passwortabfrage zu wiederholen. Die zweite Schnittstelle ist die Socket-Verbindung des Security-Servlets zum Server-Plugin. Die Kommunikation zwischen diesen beiden Komponenten läuft über Zeichenketten ab. Will ein User des Systems ein geschütztes Dokument einsehen, so zerlegt das Server-Plugin die URI der Anfrage in zwei Teile: den Sessionkey und den Rest der URI, der beschreibt, um welches Dokument es sich handelt. Diese beiden Zeichenketten werden dann über die Socket-Verbindung an das Security-Servlet geschickt. Das Servlet sendet dann nach Überprüfung der Daten eine entsprechende Antwort an das Servlet zurück:

- **ok**: Die Daten wurden überprüft und der User darf auf das angeforderte Dokument zugreifen.

- **standard + HashID**: Die Lizenz des Users ist zwar gültig, aber der Sessionkey ist abgelaufen. Das Server-Plugin sendet nun ein Formular mit einer Passwortabfrage an den User. Bei korrekter Eingabe des Kennworts wird ein neuer Sessionkey erstellt. Die HashID dient zum Erhalt der Daten des Servlets während der Abfrage.
- **firstClick + HashID**: Die Lizenz des Users ist zwar gültig, allerdings wurde zum ersten Mal auf das Dokument zugegriffen. Daher sendet das Server-Plugin ein Formular mit einer Passwortabfrage an den User. Wurde das Kennwort richtig eingegeben, so wird ein gültiger Sessionkey erstellt. Die HashID dient zum Erhalt der Daten des Servlets während der Abfrage.
- **lizenzabgelaufen**: Die Lizenz des Users ist ungültig. Daher wird der Zugriff auf das Dokument verweigert.
- **nochnichtbezahlt**: Der User hat zwar eine korrekte Lizenz für das Dokument, jedoch ist die Bezahlung noch nicht erfolgt.
- **BadFitFehler**: Der User hat keine Lizenz für das angeforderte Dokument oder der Sessionkey ist fehlerhaft. Daher wird der Zugriff auf das Dokument verweigert.

Erhält das Server-Plugin eine Zeichenkette, die nicht oben aufgeführt ist, so wird die Bearbeitung der Anfrage abgebrochen. Dann wird eine Meldung an den Client gesendet, die auf den Fehler im Server-Plugin hinweist.

5.7 Erweiterbarkeit der vorhandenen Sicherheitstechniken

Eine Verfeinerung des im System integrierten Lizenz-Konzepts könnte dadurch erreicht werden, dass der Schnittstelle zwischen Security-Servlet und Server-Plugin neue Zeichenketten hinzugefügt würden. So könnte beispielsweise der in der Schnittstellenbeschreibung genannte „BadFitFehler“ noch weiter präzisiert werden, um den User besser darüber zu informieren, warum der Zugriff auf das Dokument verweigert wurde.
Ein weiterer Ansatzpunkt für eine Erweiterung der Dokumentenschutz-Komponente ist die befehlsorientierte Arbeitsweise der Socketkommunikation. Denn bevor das Server-Plugin Daten an das Security-Servlet sendet, übergibt es eine Befehls-Zeichenkette an das Servlet. Diese Befehls-Zeichenkette besteht zur Zeit nur aus einem „a“, da außer der Überprüfung der Lizenz keine weiteren Sicherheitstechniken umgesetzt wurden. Sollen dem System weitere Sicherheitskonzepte hinzugefügt werden, so müsste lediglich der Befehlssatz erweitert werden. Außerdem müsste im Security-Servlet eine Methode implementiert werden, die den neuen Befehl abarbeitet. So wäre es möglich, unterschiedliche Lizenztypen wie Gleitlizenzen oder Gruppenlizenzen einzuführen, die schon im Server-Plugin entsprechend behandelt würden. Es wird deutlich, dass eine Erweiterung der Sicherheitstechniken mit einer Erweiterung der Schnittstelle einhergeht. Sie ist der Hauptansatzpunkt, falls neue Sicherheitskonzepte in das System integriert werden sollen.

5.8 Das Problem bei der Umbenennung der Dateinamen

Das Ziel der Sicherheitsmaßnamen des Webshops ist es, dem User es so schwer wie möglich zu machen, das Lizenzkonzept zu umgehen und damit um das Zahlen von Lizenzgebühren für Dokumente im Webshop herumzukommen. Die einfachste Möglichkeit, so wenig Geld wie möglich für eine Lizenz auszugeben ist es, eine kurze Lizenz zu kaufen und dann innerhalb der Lizenzdauer alle Daten, die zu einem Dokument gehören, lokal auf den eigenen Rechner zu speichern, damit danach keine Lizenz mehr nötig ist, um das Dokument zu lesen. Dies ist z.B. mit dem Programm wget möglich, wird allerdings auch mittlerweile von mehreren Browsern und interaktiven Robot-Programmen zur Verfügung gestellt.
Was würde man durch das Umbenennen der Dateinamen eines Dokuments gewinnen? Um dies zu verstehen, muss man sich mit dem Aufbau von HTML-Dateien, in denen die Text-Daten abgelegt sind, beschäftigen. Eine HTML-Datei enthält zwei wichtige Bestandteile, die den User interessieren. Das erste ist der textuelle bzw. graphische Inhalt, also das, wofür er Lizenzkosten bezahlt. Das zweite sind sogenannte Links, Verknüpfungen über die der User innerhalb eines Dokuments zu unterschiedlichen Stellen springen kann. Diese Links können zwar auf eine einzige HTML-Datei beschränkt sein, aber für gewöhnlich handelt es sich dabei um mehrere Dateien, die über Links miteinander verknüpft sind. In diesen Links stehen die Namen der HTML-Dateien, die geladen werden, wenn auf diese Links geklickt wird. Ändert man die Namen der HTML-Dateien, so sind die Links unbrauchbar, weil es zu ihnen keine korrespondierenden HTML-Dateien mehr gibt.
Das Konzept war, dass der User über den Klick auf einen Link eine HTML-Datei anfordert, der Name dieser Datei vom Webshop-System auf den umbenannten Namen umgesetzt wird und dem User dann die richtige Datei mit einem falschen Namen gesendet wird. Dadurch würde er keine Einschränkungen erleiden und im Zweifelsfall noch nicht einmal etwas davon mitbekommen. Für den Fall, dass er sich die angeforderten Dateien lokal auf seinem Rechner speichert, wäre er nicht mehr in der Lage, auf dem Dokument zu navigieren, weil alle Links unbrauchbar wären. Ihm

fehlt dann in diesem Fall das Webshop-System als Übersetzer der Dateinamen.
Für dieses Konzept gab es mehrere Ansätze. Der erste Ansatz war das Apache-Modul Modrewrite. Modrewrite ermöglicht es intern, die Namen von angeforderten Dateien umzubenennen, und dem User daraufhin Dateien mit anderen Dateien zu senden. Dieses Verfahren ist sehr effektiv auf Webservern, wo sich die Dateien auf dem Webserver oft ändern, jedoch nicht die Einstiegsseite bzw. die Links, die man im Internet, z.B. in Suchmaschinen, findet, die auf diesen Webserver verweisen.
Um mit Modrewrite auf Anfragen des Users zu reagieren, werden Perl-Skripte benutzt, die den Namen einer angeforderten Datei identifizieren und einen entsprechenden Namen an den Webserver weitergeben, welcher dann die Datei mit dem neuen Namen an den Browser des User sendet. Das Problem mit Modrewrite ist jedoch, dass diese Software darauf ausgelegt ist, den User nicht über Änderungen am Webserver zu informieren, der User bekommt davon nichts mit. Die Überlegung der Projektgruppe war daraufhin, die Aufgabe vom Security-Servlet und dem C-Modul übernehmen zu lassen, indem eine entsprechende Umrechnung von dort aus durchgeführt wird. Das hat zu interessanten weiteren Problemen geführt, nämlich der Möglichkeit, dass der User eine Datei anfordert, von der er schon eine Version mit umbenannten Dateinamen hat. Fordert er unter diesem Namen per Reload die Datei neu an, muss der Webserver erkennen, dass es sich bei dem Dateinamen schon um eine umbenannte Datei handelt. Sollte er die Umbenennung nicht erkennen, wird er keine korrespondierende Datei finden, die er dem User schicken könnte. Die Dateinamen müssen eindeutig erkennbar sein. Dies kann man jedoch nicht im Dateinamen selbst kodieren, weil immer die Möglichkeit besteht, dass jemand eine Datei genau so benennt, wie der Webshop sie umbenennt. Deswegen war ein weiterer Eintrag in der Datenbank geplant, in dem der umbenannte Dateiname steht. Darüber hätte es eine eindeutige Identifizierung gegeben.
Leider wurde zuviel Zeit in den Entwurf dieser Sicherheitstechnik investiert, weil eine Realisierung mit dem Apache Webserver nicht möglich erscheint. Das liegt daran, dass vom C-Modul Dateien per Sub-Request an den User geschickt werden. Diese Sub-Requests ermöglichen den Informationsfluß zwischen C-Modul und Security-Servlet mit der Entscheidungsfähigkeit, ob eine Datei an den User gesendet wird, oder nicht. Bei einem Sub-Request wird jedoch nichts anderes als bei dem Modul Modrewrite gemacht, nämlich eine Datei, mit einem beliebigen Namen, an den Browser des User gesendet. Der User bekommt davon nichts mit, weil sein Browser darüber nicht informiert wird, weswegen er die Datei unter dem Namen erhält, unter dem sie angefordert wurde. Das bedeutet, dass nichts dadurch gewonnen werden kann.
Eine kurzfristige Alternative schien das Arbeiten mit Meta-Refresh-Aufrufen zu sein, weil über diesen Weg dem User Dateien mit neuen Dateinamen gesendet werden können. Das liegt daran, dass bei dieser Methode der User keinen Einfluss darauf hat, was ihm geschickt wird. Allerdings wird für jeden Meta-Refresh-Aufruf eine eigene HTML-Datei benötigt, weswegen auch dieses Verfahren nicht funktionieren kann. Dadurch würde nämlich der User eine Datei unter dem richtigen Namen, in der sich ein Meta-Refresh-Aufruf befindet, geschickt bekommen, welche ihm eine zweite Datei schickt, in der die eigentlichen Daten stehen, welche einen falschen Dateinamen hat. Er bekommt also doppelt so viele Dateien zugeschickt, kann aber genauso wie vorher lokal darauf auf seinem Rechner navigieren, weil ihn jede Datei mit richtigem Namen automatisch zu einer Datei mit falschem Namen weiterführt.
Zusammenfassend muss gesagt werden, dass es sich hier um einen Fehlschlag handelt, jedoch um einen Fehlschlag aus dem viele Erkenntnisse gewonnen werden konnten. Es wurden neue Erfahrungen mit der Programmiersprache Perl gesammelt und Einsichten über die Funktionalität des Apache-Webserver gewonnen. Auch die Problematik der nicht eindeutigen Dateinamen hat zu neuen Ideen in der Nutzung von Datenbanken geführt, was also auf lange Sicht hin als kapitales

Wissen angesehen werden kann.

Kapitel 6

Authoring - Komponente

6.1 Entwurf

6.1.1 Designentscheidung

Das zentrale Objekt des Webshops ist das Dokument. Zu einem Dokument gehören eine Vielzahl von Informationen, die in irgendeiner Weise auf der Ebene der Programmiersprache repräsentiert werden müssen. Die Darstellung dieser Informationen und deren Beziehungen zueinander sind ein wesentlicher Teil des Designs. Die Frage nach dem besten Design wird durch ein Abwägen von Verständlichkeit, Erweiterbarkeit und Korrektheit beantwortet. Dem Webshop liegt als Design das Holderkonzept zugrunde, das im Folgenden ausführlich beschrieben wird. Darüber hinaus werden alternative Möglichkeiten aufgezeigt, die Gegenstand früherer Diskussionen waren.

6.1.1.1 Das Holderkonzept

Die eigentliche Idee des Holderkonzeptes ist es, zusammengehörige Informationen eines Dokumentes innerhalb einer gemeinsamen Klasse zu kapseln. Diese Bündelung von vielen kleinen Informationen erzeugt eine überschaubare Menge von grösseren Informationspaketen. Zusätzlich übernehmen Holder gewisse Aufgaben, die ein konsistentes Verarbeiten der Informationen garantieren. Da alle Holder diese Methoden implementieren müssen, ist eine einheitliche Arbeitsweise mit verschiedenen Holdern möglich. Verwaltet werden die Holder von dem Dokument, das die Holder setzt bzw. wieder entfernt. Diese abstrakte Sichtweise des Holderkonzeptes wird durch die Bildung eines konkreten Holders verständlich. Zu einem Dokument gehören Informationen wie ein Titel, eine Beschreibung und ein Verlag. Diese drei Informationen werden zu einer Klasse *ThreeInfosHolder* zusammengefasst, indem sie als Attribute in der Klasse vorkommen. Ein Zugriff auf die einzelnen Informationen geschieht über die Klasse *ThreeInfosHolder*, beispielsweise mittels Get- und Set-Methoden. Zusätzlich übernimmt die Klasse die Aufgabe, die drei Informationen bei Bedarf in einer Datenbank zu speichern. Ein Dokument besteht nun nicht mehr in erster Linie aus Titel, Beschreibung und Verlag, sondern aus einem *ThreeInfosHolder* und kann die Werte der drei Informationen über den Aufruf der Speicher-Methode des Holders bequem in der Datenbank speichern. Nach dem gleichen Prinzip werden weitere Holder gebildet bis alle Informationen einem Holder zugeordnet sind und somit ein Dokument nur noch aus Holdern besteht. Sollen nun alle Informationen in der Datenbank gespeichert werden, muss bei

allen Holdern nur die Speicher-Methode aufgerufen werden und dies kann einheitlich innerhalb einer Schleife erfolgen.

6.1.1.2 Die Datenstruktur

Jede Klasse, die das Interface *Holder* implementiert, ist vom Typ Holder. Das Interface enthält zwei Methoden, eine Schreibmethode 'write(int Document)' und eine Reinitialisierungsmethode 'rollBack(int Document)', wobei das Argument 'Document' die in der Datenbank verwendete Identifikationsnummer eines Dokumentes darstellt. Die Schreibmethode soll jeder Holder so implementieren, dass die eigentlichen Inhalte des Holders in der Datenbank gespeichert werden. Die Reinitialisierungsmethode soll den Holder mit den zuletzt in der Datenbank gespeicherten Werten füllen. Bei den Datenbankzugriffen können Exceptions auftreten, die von den beiden Methoden des Interfaces weitergegeben werden und somit von anderen Methoden abgefangen werden müssen. Die Motivation für die Realisierung des Holders mittels eines Interfaces ist die Flexibilität der Datenstruktur. Im Webshop werden die zu einem Dokument gehörenden Informationen auf vier Holder verteilt, *DetailHolder*, *AuthorHolder*, *OfferHolder* und *UrlHolder* (siehe Abbildung 6.1).

1. Der *DetailHolder* kapselt alle Informationen eines Dokumentes, die eine allgemeine Beschreibung des Dokumentes darstellen. Hierzu gehören unter anderem der Titel, die Beschreibung, die ISBN, der Verlag und das Erscheinungsjahr. Der *DetailHolder* ist eine direkte Unterklasse von *Object* und ermöglicht einen Zugriff auf die Attribute mittels Get- und Set-Methoden.
2. Der *AuthorHolder* enthält alle Autoren zu einem Dokument.
3. Der *OfferHolder* enthält alle Angebote zu einem Dokument.
4. Der *UrlHolder* enthält alle Urls zu einem Dokument.

Die letzten drei Holder unterscheiden sich von dem *DetailHolder* konzeptionell dadurch, dass sie eine Unterklasse der Klasse *Vector* sind und somit alle Methoden zum Zugriff auf die enthaltenen Elemente von der Oberklasse erben. Es lassen sich also zwei Typen von Holder bei dieser Realisierung des Webshops unterscheiden: Holder mit Informationen, die häufig im gleichen Kontext stehen aber einen unterschiedlichen Typ besitzen und Holder, die eine Liste von gleichen Informationstypen darstellen. Andere Realisierungen sind durchaus denkbar und bei einer Erweiterung des Webshops eventuell sogar nötig.

6.1.1.3 Das Holderkonzept im GUI-Klassenmodell

Die Darstellung der Holder-Informationen erfolgt in jeweils eigenen Fenstern, die jedoch gewisse Funktionalitäten gemeinsam haben. Daher gibt es eine Klasse *HolderFrame*, die die Basis für alle Holder-Fenster darstellt (siehe Abbildung 6.2). Diese Klasse enthält zwei Buttons, einen zum Speichern des Inhaltes in der Datenbank und einen zum Reinitialisieren des Inhaltes mit den zuletzt gespeicherten Daten. In jeder Instanz dieser Klasse *HolderFrame* befindet sich für einen Informationsbereich des Dokumentes eine Erweiterung der abstrakten Klasse *HolderPanel*, welche grundlegende Funktionalitäten kapselt, die jedes Panel in einem *HolderFrame* benötigt.

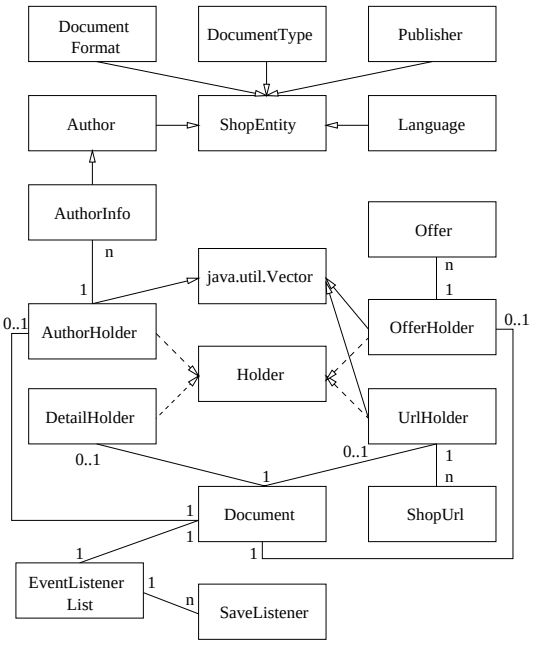

Abbildung 6.1: Das allgemeine Klassenmodell des GUI

Es existiert also für jeden Holder-Typen ein eigenes Fenster, das die Informationen eines konkreten Holders in einem dem Typen entsprechenden *HolderPanel* darstellt. Um beispielsweise die Informationen eines *DetailHolder* anzuzeigen, wird eine Klasse *DetailHolderPanel* benutzt, die eine Unterklasse von *HolderPanel* ist. Im *DetailHolderPanel* werden die benötigten Textfelder, Tabellen und Ähnliches erzeugt und positioniet, um die Inhalte eines *DetailHolder* anzuzeigen. Dieses *DetailHolderPanel* ist in einem *HolderFrame* eingebettet und wird von diesem angezeigt und verwaltet.

6.1.1.4 Vorteile des Holderkonzeptes

Ein wesentlicher Punkt beim Entwurf des GUI-Klassenmodells war das dynamische Hinzufügen und Entfernen von Informationen zu bzw. von einem Dokument. Nur die zur gewünschten Arbeit notwendigen Daten sollten aus der Datenbank geladen werden und umgekehrt sollten bereits korrekte Daten auf Wunsch sofort in die Datenbank geschrieben werden. Ein Dokument besteht nun aus vier Holder, die jeweils verwandte Informationen enthalten. Verwandt soll hier zum Ausdruck bringen, dass diese Informationen häufig im selben Kontext erscheinen und ein Bearbeiten einer Informationseinheit im Zusammenhang mit den anderen Informationen geschieht. Ein Holder ist also ein Kompromiss aus Dynamik und Ergonomie. Dynamisch, weil nicht alle zu einem Dokument gehörenden Informationen pauschal geladen werden und ergonomisch, weil nicht jede Informationseinheit einzelnd geladen wird und somit dem Benutzer ein ausdrückliches Anfordern der nächsten Information erspart wird. Je nach Aufgabe werden also nur die benötigten Informationsbereiche geladen bzw. gespeichert.

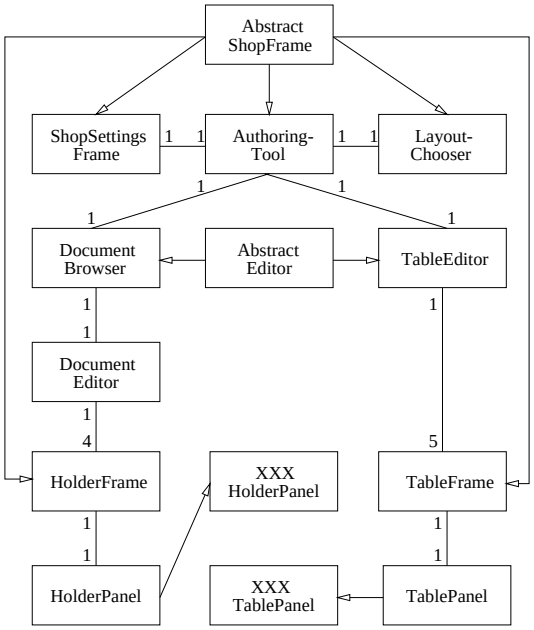

Abbildung 6.2: Das spezielle Klassenmodell des GUI

6.1.1.5 Alternativen zum Holderkonzept

Anstatt die kennzeichnenden Elemente eines Dokumentes auf Holder aufzuteilen, hätten sie auch direkt als Attribute in einem Dokument dargestellt werden können. Folge wäre eine recht unübersichtliche Klasse *Document*, die direkt auf die Attribute zugreifen könnte. Der Vorteil hierbei ist jedoch, dass es keine Bündelung verschiedener Informationen gibt, die eine gemeinsame Bearbeitung nahelegt. Es ist offen, ob der Titel zusammen mit dem Verlag in einem Panel bearbeitet wird oder ob beide Informationen in getrennten Panels angezeigt werden. Sicherlich ist diese Offenheit auch mitels der Holder zu realisieren, jedoch wird dies bei einer nicht völligen Überarbeitung der Holder mit einem gewissen Overhead einhergehen. Das Holderkonzept ist darauf ausgerichtet, die Inhalte eines Holders zusammen in einem Panel anzuzeigen. Bei den meisten Holdern ist jedoch abzusehen, dass ihre Inhalte nicht geändert werden, wie z.B. bei allen Holdern, die Listen von Objekten gleichen Typs enthalten, nämlich *AuthorHolder*, *OfferHolder* und *UrlHolder*. Ebenfalls denkbar ist eine Mischform aus Holder und einzelnen Dokumentattributen. Auf den ersten Blick würden die Vorteile des einen Konzeptes mit den Vorteilen des anderen Konzeptes kombiniert werden. Nachteil ist jedoch hierbei, dass die einheitliche Behandlung bei gewissen Aufgaben wie dem Speichern verloren geht, bzw. auf einer niedrigeren Stufe für alle Attribute ermöglicht werden muss.

6.1.2 Entwurf der Benutzer-Schnittstelle

Der Entwurf der Benutzer-Schnittstelle teilt sich in die Bereiche Ablaufdiagramme und Maskenentwürfe auf.
Die Ablaufdiagramme wurden auf der Grundlage der Designentscheidung festgelegt. Sie berücksichtigen bereits das Holder-Konzept und stellen die zur Umsetzung benötigten Teilaufgaben

und deren Reihenfolge dar.
Die Maskenentwürfe für die Benutzungsoberfläche wurden ebenfalls unter Berücksichtigung des Holder-Konzepts erstellt. Die Skizzen sollen die Repräsentation der Daten und der Interaktionselemente verdeutlichen.

6.1.2.1 Ablaufdiagramme

Es wurden Ablaufdiagramme für folgende wesentliche Funktionalitäten der Benutzungsoberfläche erstellt:

- Die Shop-Einstellungen können vom Benutzer geändert werden (Abbildung 6.3)
- Der Benutzer kann ein neues Dokument anlegen (Abbildung 6.4)
- Er kann ein Dokument löschen (Abbildung 6.5)
- Er kann die zum Dokument gehörenden Informationen ändern (Abbildung 6.6)

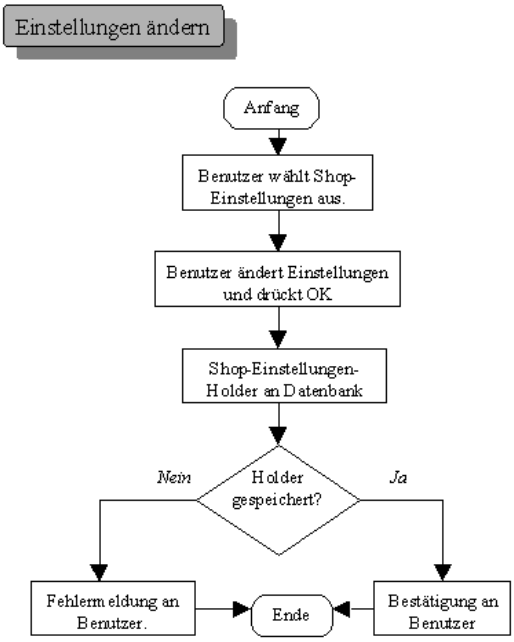

Abbildung 6.3: Shop-Einstellungen ändern

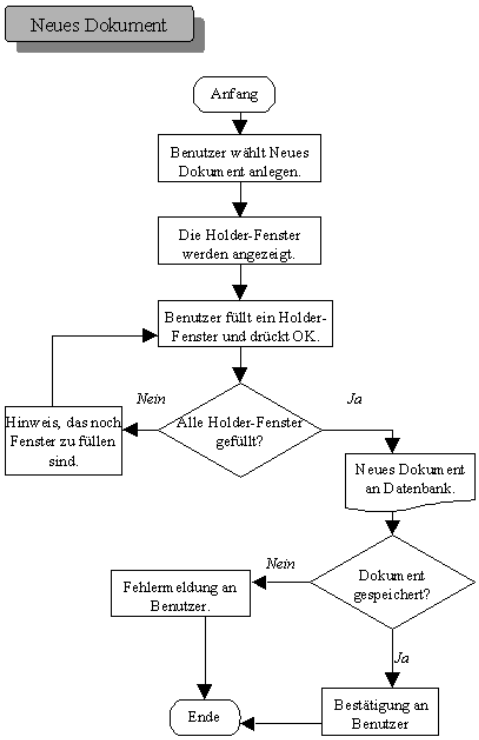

Abbildung 6.4: Neues Dokument

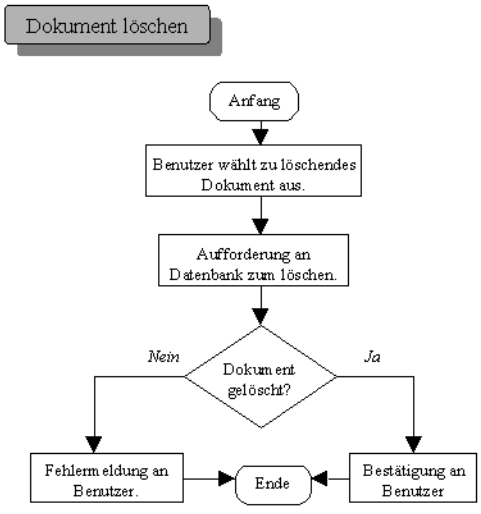

Abbildung 6.5: Dokument löschen

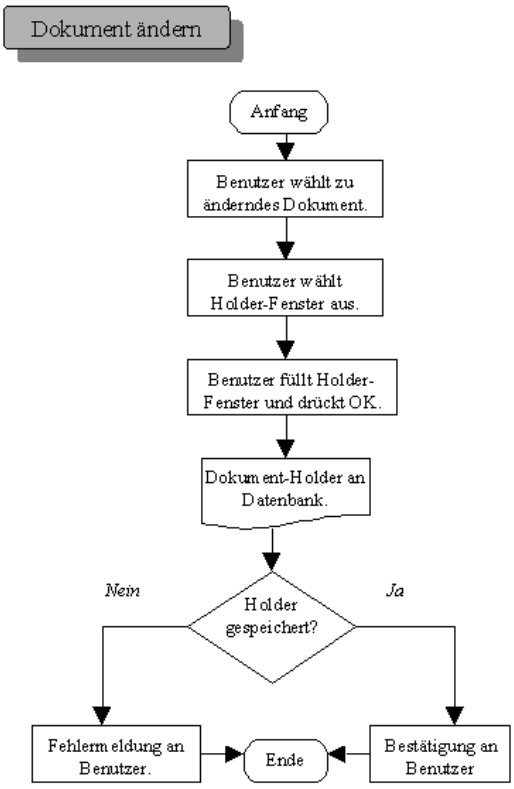

Abbildung 6.6: Dokument ändern

6.1.2.2 Maskenentwurf

Die Oberfläche des Authoring-Tools lässt sich in drei Bereiche aufteilen: der Objekterstellungs-Bereich, der Tabellen-Bereich und der Einstellungs-Bereich. Die einzelnen Bereiche werden in den folgenden Unterkapiteln näher beschrieben. Anzumerken ist, dass das Layout der Masken lediglich ein Entwurf ist. Das endgültige Aussehen der Masken kann vom Entwurf selbstverständlich variieren.

- Der Objekterstellungs-Bereich

  Die Oberfläche des Authoring-Tools ist im Dateimanager-Stil gehalten. Jedes Objekt in der Datenbank, welches über das Internet angeboten wird, ist durch ein kleines Symbol in der Hauptansicht (Abb. 6.7) charakterisiert. Dabei wird versucht, das dem Objekt zugeordnete Bild zu verwenden. Falls dies nicht möglich ist, wird ein Standard-Bild angezeigt. Die linke Seite der Hauptansicht stellt eine Baumstruktur dar, welche die die Klassifizierung der Objekte in der Datenbank wiederspiegelt. Die gesamte Oberfläche benutzt Drag-und-Drop im Sinne eines weit verbreiteten Dateimanagers für ein bekanntes Fenster-Betriebsystem.

  Die einzelnen Informationen zu einem Objekt (Angebote, Urls etc.) werden in verschiedenen Fenstern eingestellt. Diese Fenster sind semantisch voneinander getrennt, d.h. es gibt jeweils ein Fenster für eine klar umrissende Informationseinheit des Obkjektes (z.B. die Dublin-Core-Informationen oder auch alle Url-Informationen des Objektes). Die einzelnen

Fenster können über das Menü 'Ansicht', das Menü 'Bearbeiten', über ein Popup-Menü, oder über die Schnellansicht unten rechts im Hauptfenster geöffnet, bzw. geschlossen werden. Sollte im Hauptfenster ein Wechsel des aktuell bearbeiteten Objektes erfolgen, werden die dargestellten Informationen in den einzelnen Fenstern aktualisiert.

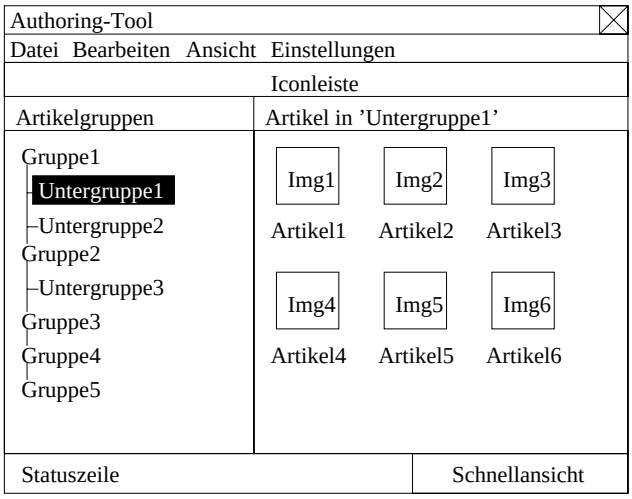

Abbildung 6.7: Authoring-Tool: Hauptfenster mit Objekten

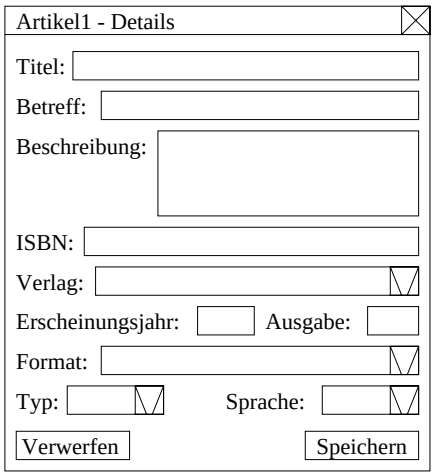

Abbildung 6.8: Authoring-Tool: Detail-Informationen zu einem Objekt

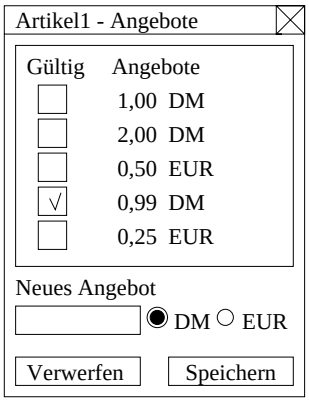

Abbildung 6.9: Authoring-Tool: Angebote zu einem Objekt

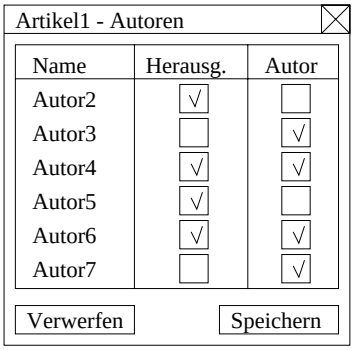

Abbildung 6.10: Authoring-Tool: Autoren zu einem Objekt

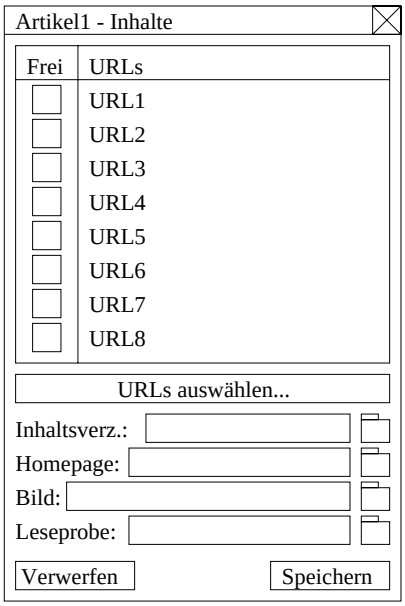

Abbildung 6.11: Authoring-Tool: URLs zu einem Objekt

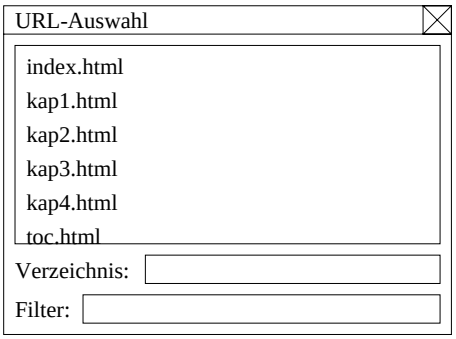

Abbildung 6.12: Authoring-Tool: Auswahl neuer URLs (Std. Filechooser)

- Der Tabellen-Bereich

  Im Tabellen-Bereich werden verschiedene Tabellen in der Datenbank gewartet. Es gibt fünf Arten von Tabellen, die alle über ein eigenes Fenster bearbeitet werden können. Die Tabellen dienen als Datengrundlage für die Erstellung neuer Objekte. Bevor z.B. ein Autor einem Objekt zugeordnet werden kann, muss er über das Fenster zur Verwaltung der Autoren-Tabelle in die Datenbank eingefügt worden sein. Die einzelnen Tabellen sind über das Menü 'Bearbeiten' und das Menü 'Ansicht' zu erreichen.

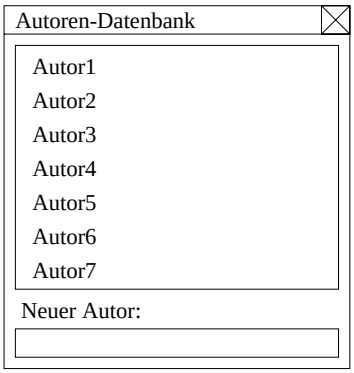

Abbildung 6.13: Authoring-Tool: Autoren-Tabelle in der Datenbank

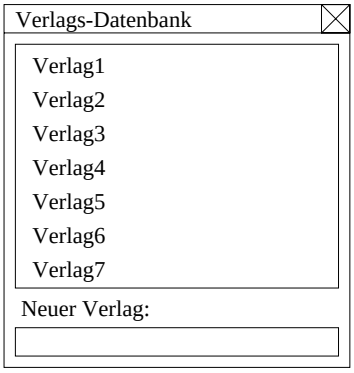

Abbildung 6.14: Authoring-Tool: Verlage-Tabelle in der Datenbank

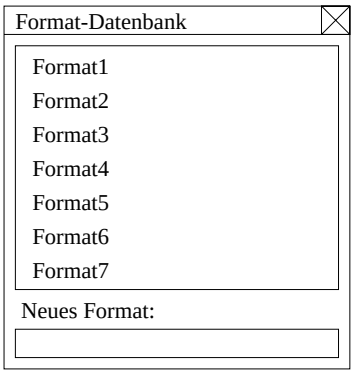

Abbildung 6.15: Authoring-Tool: Formate-Tabelle in der Datenbank

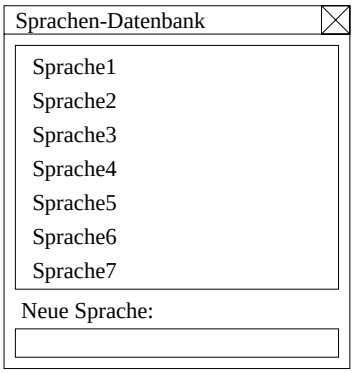

Abbildung 6.16: Authoring-Tool: Sprachen-Tabelle in der Datenbank

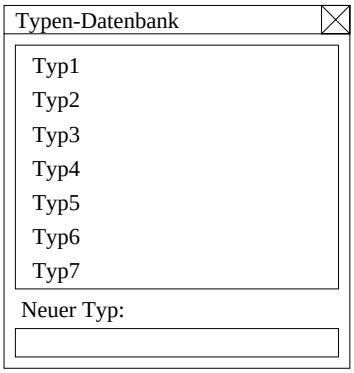

Abbildung 6.17: Authoring-Tool: Typen-Tabelle in der Datenbank

- Der Einstellungs-Bereich

  Der Einstellungs-Bereich unterteilt sich in vier Teilbereiche: Allgemeine Einstellungen (Abb 6.18), Internet-Einstellungen (Abb. 6.19, Payment-Einstellungen (Abb. 6.20) und Einstellungen zum verwendeten Layout (Abb. 6.21). Die ersten drei Bereiche befinden sich ein einem Fenster und sind durch Register voneinander getrennt. Das Fenster für die Layout-Einstellungen ist aufgrund der Größe der einzelnen Vorschau-Bilder in ein separates Fenster ausgelagert.Bei der Layout-Auswahl kann per Doppelklick auf ein Layout eine vergrößerte Vorschau des Layouts gewählt werden (Abb. 6.22).

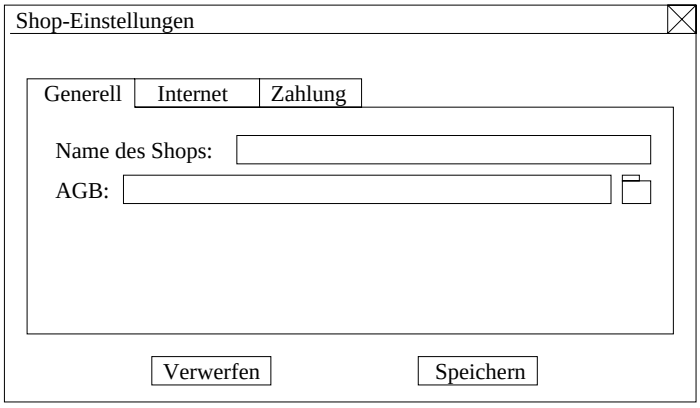

Abbildung 6.18: Authoring-Tool: Allgemeine Shop-Einstellungen

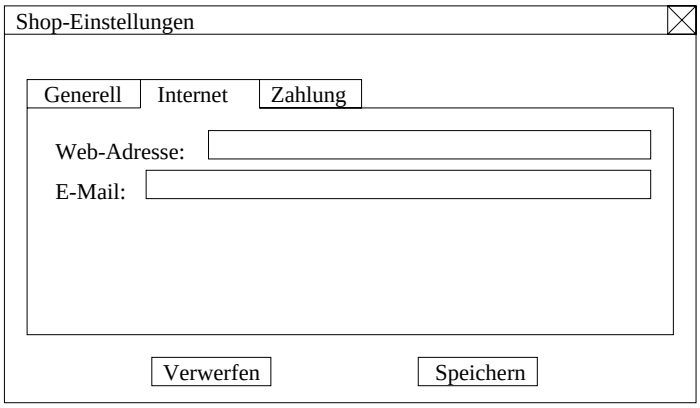

Abbildung 6.19: Authoring-Tool: Internet-Einstellungen des Shops

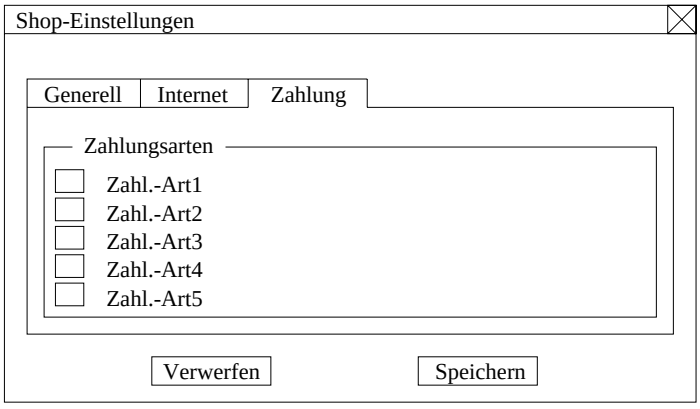

Abbildung 6.20: Authoring-Tool: Payment-Einstellungen des Shops

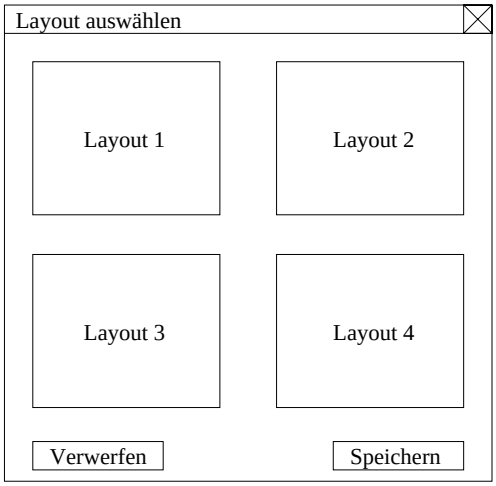

Abbildung 6.21: Authoring-Tool: Auswahl eines Layouts

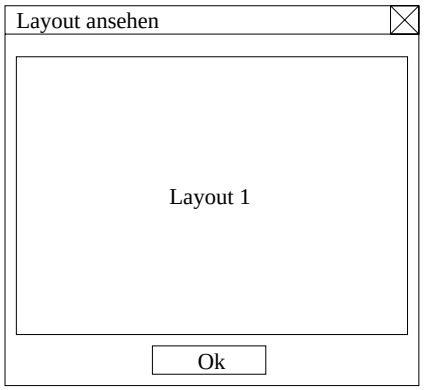

Abbildung 6.22: Authoring-Tool: Detailansicht eines Layouts

6.2 Implementierung

Nachdem der Entwurf des Authoring-Tools festgelegt war, startete die Implementierungs-Phase des Programms, welche rund zwei Monate dauerte. Während dieser Zeit waren die Autoren des Tools oft gezwungen, den Entwurf ein klein wenig abzuändern, da einige Teilaspekte des Entwurfes gar nicht oder nur sehr schwierig realisierbar erschienen. Darüber hinaus waren die Autoren des Tools oft gefordert, spezielle Detail-Lösungen für besondere Probleme zu finden. Obendrein traten auch kleinere Fehler in der verwendeten Java-Entwicklungsumgebung in der Version 1.2.2 zu Tage, für die in einigen Fällen ein *Workaround* entwickelt wurde, in anderen Fällen aber keine Lösung vorhanden war.

Zu guter Letzt mussten nach Abschluss der Implementierungs-Phase noch einige Modifikationen am fertigen Quell-Code vorgenommen werden, da bei Tests des fertigen Programms auf verschiedenen Plattformen ein unterschiedliches Verhalten zu Tage trat, was die Entwickler des Authoring-Tools an der Plattformunabhängigkeit der Java-Programmiersprache zweifeln ließ.

Die endgültige Programmierschnittstelle des Authoring-Tools und einige Detail-Lösungen werden in des folgenden Unterkapiteln präsentiert.

6.2.1 Schnittstellen-Dokumentation

Das Authoring-Tool selber hat keine besonderen Schnittstellen, die von anderen Anwendungen benutzt werden könnten. Stattdessen gibt es einige abstrakte Klassen, auf denen das Authoring-Tool aufbaut. Diese Klassen stellen sozusagen das Grundgerüst des Authoring-Tools dar und sollen im Folgenden erklärt werden. Weitere Informationen zu den Klassen des Authoring-Tools stehen in der HTML-Dokumentation.

Klasse AbstractShopFrame: Der AbstractShopFrame ist die Oberklasse aller Fenster des Authoring-Tools. Er stellt die Möglichkeit bereit, *StatusListener* zu registrieren und diesen Status-Meldungen zukommen zu lassen. Desweiteren bietet er die Funktion, J-Komponenten in Höhe und Breite miteinander zu synchronisieren. Die Methoden der Klasse lauten:

- *void addStatusListener()* registriert einen *StatusListener*, damit dieser Status-Meldungen dieses Fensters erhält.
- *void removeStatusListener()* deregistriert einen *StatusListener*, damit dieser von diesem Fenster keine Status-Meldungen mehr erhält.
- *JComponent[] syncAllWidths()* synchronisiert die Breite von J-Komponenten.
- *JComponent[] syncAllHeights()* synchronisiert die Höhe von J-Komponenten.
- *void fireStatusChanged()* benachrichtigt alle registrierten *StatusListener*, dass sich der Status dieses Fensters geändert hat.

Klasse HolderPanel: Das HolderPanel ist die Oberklasse aller JPanel, die die Daten der verschiedenen Holder aus dem webshop.classes-Package repräsentieren. Die Methoden der Klasse lauten:

- *void writeHolder()* speichert den Daten des repräsentierten Holders in die Datenbank.

- *boolean isHolderSet()* zeigt an, ob das JPanel aktuell einen Holder repräsentiert.
- *setDocument()* setzt das Dokument, dessen Holder angezeigt werden soll. Sobald das Dokument gesetzt ist, wird der entsprechende Holder aus der Datenbank geladen.
- *Document getDocument()* liefert das Dokument zurück, dessen Holder aktuell angezeigt wird.
- *void setHolder()* setzt den Holder, dessen Daten durch das JPanel repräsentiert werden.
- *Holder getHolder()* liefert den Holder, dessen Daten durch das JPanel repräsensiert werden.
- *void refreshView()* veranlasst das JPanel, die angezeigten Daten durch die Daten des zu repräsentierenden Holders zu ersetzen.
- *void prepareHolder()* veranlasst das JPanel, den Holder mit den angezeigten Daten zu aktualisieren.
- *Holder requestHolder()* lädt den durch das JPanel repräsentierten Holder aus der Datenbank.
- *void setEnabled()* setzt den Bearbeitungs-Status des JPanels.

Klasse TablePanel: Das TablePanel ist die Oberklasse aller JPanel, die eine Tabelle in der Datenbank repräsentieren. Die Methoden der Klasse lauten:

- *JList getList()* liefert die eine Liste mit den Elementen der repräsentierten Datenbank-Tabelle.
- *boolean addNewElement()* fügt ein neues Element in die repräsentierte Datenbank-Tabelle ein.

Klasse AbstractEditor: Der AbstractEditor ist die Oberklasse aller Editoren im Authoring-Tool. Editoren sind Klassen, die Editier-Fenster verwalten, und diese bei Bedarf öffnen bzw. schließen, falls ein Editier-Fenster (nicht mehr) erforderlich ist. Die Methoden der Klasse lauten:

- *void addStatusListener()* registriert einen *StatusListener*, damit dieser bei Status-Änderungen dieses Editors benachrichtigt wird.
- *void removeStatusListener()* deregistriert einen *StatusListener*, damit dieser keine Status-Meldungen von diesem Editor mehr erhält.
- *void fireStatusChanged()* benachrichtigt alle registrierten *StatusListener*, dass eine Status-Änderung aufgetreten ist.
- *boolean addVisibilityListener()* verknüpft eine *ButtonModel* mit einem Schlüssel, damit dieses Modell gegebenenfalls benachrichtigt wird, falls eine Sichtbarkeits-Änderung eines Editor-Fensters aufgetreten ist.
- *boolean removeVisibilityListener()* entfernt die Verknüpfung zwischen einem Schlüssen und einem *ButtonModel*, damit dieses nicht mehr bei Sichtbarkeits-Änderungen benachrichtigt wird.
- *notifyVisibilityListeners()* benachrichtigt alle Button-Modelle, die einem übergebenen Schlüssel zugeordnet sind, ob das diesem Schlüssel zugeordnete Fenster aktuell sichbar ist.

Klasse AbstractThumbnailModel: Das AbstractThumbnailModel stellt das zugrundeliegende Modell eines J-Thumbnails dar. J-Thumbnails sind die kleinen Symbole, mit der die Dokumente in der Datenbank repräsentiert werden. Die Methoden der Klasse lauten:

- *void addChangeListener* registriert einen *ChangeListener*, der bei Änderungen am Modell benachrichtigt werden soll.
- *void removeChangeListener* deregistriert einen *ChangeListener*, damit dieser nicht mehr bei Änderungen am Modell benachrichtigt wird.
- *void fireStateChanged* benachrichtigt alle registrierten *ChangeListener*, dass eine Änderung am Modell aufgetreten ist.
- *void setTitle* setzt den Titel des Modells und benachrichtigt gegebenenfalls alle registrierten *ChangeListener*.
- *String getTitle* liefert den Titel des Modells.
- *void setIconFile()* setzt den Pfad zu einem Bild, welches von einem J-Thumbnail angezeigt werden soll und benachrichtigt gegebenenfalls alle registrierten *ChangeListener*.
- *String getIconFile()* liefert den Pfad zu einem Bild.
- *void setEnabled()* setzt die Benutzbarkeit des J-Thumbnails.
- *boolean isEnabled()* liefert die Benutzbarkeit des J-Thumbnails.
- *void setSelected()* setzt den Selektions-Status des J-Thumbnails und benachrichtigt gegebenenfalls alle registrierten *ChangeListener*.
- *boolean isSelected()* zeigt an, ob das J-Thumbnail aktuell selektiert ist.
- *void setScaling()* setzt die Art und Weise, mit der das Bild skaliert werden soll und benachrichtigt gegebenenfalls alle registrierten *ChangeListener*. Der Parameter muss eine Skalierungs-Konstante der Klasse *AbstractThumbnailModel* sein.
- *int getScaling()* liefert die Art und Weise, mit der das Bild skaliert wird.

6.2.2 Problemlösungen

Wie bereits zu Beginn dieses Abschnitts angedeutet, gab es bei der Implementierung des Authoring-Tools immer wieder kleine Probleme, die spezielle Detail-Lösungen erforderten. Einige dieser Lösungen sollen im Folgenden dargestellt werden.

- Problem: Konsistenz von Daten in verschiedenen Fenstern

  Durch die Verwendung verschiedener voneinander getrennter Fenster für die Verwaltung der Datenbank-Tabellen *Author, Verlage, Dokument-Formate, Dokument-Typen und Sprachen* gab es ein Problem, Änderungen in diesen Tabellen anderen Programm-Bereichen - vor allem das Fenster mit den Details zu einem Artikel - mitzuteilen. Die Lösung dieses Problems bestand darin, eine Klasse zu erschaffen, die entsprechende Tabellen statisch für alle Instanzen des Programms bereit stellte. Da es sich bei den Tabellen um einfache Listen handelte, wurde zur Kapselung der Daten das *ListModel* des Swing-Paketes aus Java 1.2 verwendet. Somit war es möglich, sich bei den verschiedenen Modellen als *ListDataListener* zu registrieren, um über Änderungen in der Liste informiert zu werden. Die Folge

war schließlich, dass in allen Funktionen des Authoring-Tools auf gemeinsame Tabellen zugegriffen werden konnte, und somit die Konsistenz der Daten gewährt war. Die Klasse, die die verschiedenen Tabellen als *ListModel* bereitstellt, ist die Klasse *DBSupporter*.

- Problem: Performance-Probleme beim Anzeigen von Artikeln

  Die Shop-Einstellungen (Klasse *ShopSettings*) sind in vielen Funktionen notwendig, und normalerweise hätte es auch gereicht, die Einstellungen bei Aufruf einer Funktion jeweils einmal aus der Datenbank zu laden. Allerdings gibt es eine elementare Funktion im Authoring-Tool, wo mit dieser Vorgehensweise große Performance-Probleme aufgetreten wären: das Anzeigen von Artikeln im Hauptfenster. Jeder Artikel im Hauptfenster wird von einer Instanz der Klasse *JThumbnail* repräsentiert. Das Problem war nun, dass zum Anzeigen des Bildes zu einem Artikel der absolute Pfad-Anteil des Bildes bekannt sein musste, welcher wiederum in den Shop-Einstellungen zu finden war. Für jedes angezeigte Bild im Hauptfenster hätte somit eine neue Anfrage an die Datenbank gestartet werden müssen, was mitunter arge Performance-Probleme zur Folge gehabt hätte. Zur Vermeidung dieser Probleme werden die Shop-Einstellungen in der Klasse *DBSupporter* statisch gecached, um sie für alle Instanzen des Programms global zur Verfügung zu stellen.

- Problem: Unvollständige Implementierung des Swing-Dateimanagers

  Ursprünglich sollte der Swing-Dateimanager benutzt werden, um Urls einem Artikel zuzuordnen. Dies war jedoch nicht möglich, da der Swing-Dateimanager *JFileChooser* in der verwendeten Version 1.2.2 der Java-Entwicklungsumgebung nicht vollständig implementiert ist, und somit eine notwendigen Funktionen des Dateimanagers nicht zur Verfügung standen. Abhilfe brachte ein eigens entwickelter Dateimanager , der auf der Klasse *JDialog* basiert und dessen Funktionalität im Wesentlichen auf einer *JList* und der Klasse *File* realisiert wird.

- Problem: Drag und Drop auf Tabellen in einem JScrollPane

  Die Klasse *JTable* des Swing-Paketes implementiert das Interface *Scrollable* nicht korrekt. Falls für die Tabelle mehr Platz vorhanden ist, als sie tatsächlich braucht, berücksichtigt sie diesen nicht, falls sie in einem *JScrollPane* liegt. Die Folge war, dass Drag und Drop nur auf Bereiche der Tabelle funktionierte, in denen Daten angezeigt wurden. Dieses Verhalten war nach Meinung der Entwickler offensichtlich falsch. Abhilfe brachte eine Modifikation zweier durch das Interface *Scrollable* zu implementierenden Methoden. Angenehmer Nebeneffekt war zudem, dass die Hintergrund-Farbe von Tabellen nun einheitlich erschien.

- Problem: Absturz des Drag-und-Drop-Systems unter Motif

  Unter Motif tritt das Problem auf, dass das komplette Drag-und-Drop-System von Java in der Version 1.2 abstürzt, sobald ein *JWindow* geöffnet wird. Dieses Problem scheint zumindest bei der Verwendung von Motif aufzutreten, unter Windows gibt es ein derartiges Problem nicht. Eine Lösung dieses Problems konnte leider nicht gefunden werden, da der Fehler offensichtlich in der Implementierung des nativen Motif-UI von Sun liegt. Vielleicht ist Java in der Version 1.3 von diesem Problem nicht mehr betroffen.

- Problem: Zuordnung von Artikeln zu Ordnern

  Die Zuordnung von Artikeln zu Ordnern in der Datenbank ist eine n-zu-m-Beziehung. Damit ein neuer Artikel dem aktuell geöffneten Ordner zugeordnet werden kann, ist es notwendig, dass das UI informiert wird, sobald ein Artikel in der Datenbank neu erstellt

wird. Allerdings besitzen die Artikel, die durch die Klasse *Document* repräsentiert werden, keine Informationen über Ordner, geschweige denn über den aktuell geöffneten Ordner in dem UI. Abhilfe schaffte hier die Verwendung eines Events *SaveEvent* mit passendem Listener-Interface *SaveListener*. Sobald ein Artikel neu in der Datenbank angelegt wird, wird *vor* dem Speichern und *nach* dem Speichern ein Event ausgelöst, welches von dem UI abgefangen und weiterverarbeitet wird.

Abschließend lässt sich sagen, dass noch viele weitere kleine Detail-Lösungen gefunden werden mussten, jedoch würde deren Aufzählung den Umfang dieses Unterkapitels sprengen. Zukünftige Entwickler sind jedoch gerne aufgefordert, sich anhand der dokumentierten Quellen des Authoring-Tools neue Ideen zu holen.

6.3 Benutzungshandbuch

6.3.1 Benutzungshandbuch des Authoring-Tools

Das Authoring-Tool ist ein Werkzeug für den Administrator des WebShops. Es dient dazu, den Datenbestand des WebShops zu erstellen und zu verwalten. Mit Hilfe des Authoring-Tools können Artikel angelegt, bearbeitet und klassifiziert werden. Zur Klassifikation der Artikel dienen frei definierbare Ordner, denen die im WebShop zu präsentierenden Artikel zugeordnet werden müssen. Ferner ist es mit dem Authoring-Tool möglich, die grundsätzlichen Einstellungen des WebShops festzulegen. Die einzelnen grafischen Schnittstellen des Authoring-Tools und die notwendigen Schritte zur Erstellung und Verwaltung des Datenbestandes werden in den folgenden Unterkapiteln erläutert.

6.3.1.1 Aufruf des Tools

Voraussetzung zum Starten des Authoring-Tools ist eine Java-Entwicklungsumgebung in der Version 1.2 (sogenannt Java 2) oder eine Java-Laufzeitumgebung (JRE), ebenfalls in der Version 1.2. Bevor das Authoring-Tool gestartet werden kann, sind allerdings noch einige Vorarbeiten nötig.

Zunächst ist die Konfigurations-Datei *webshop.ini* anzupassen. Die notwendigen Einstellungen betreffen die Anbindung des Authoring-Tools zur Datenbank, sowie Informationen zum verwendeten Payment-Handler, der die Abwicklung von Transaktionen im WebShop übernimmt. Anschließend kann das Authoring-Tool mit dem Aufruf

java -Dwebshop.ini=<voller Pfad zur webshop.ini> webshop.gui.AuthoringTool

oder - im Falle der Java-Laufzeitumgebung - mit

jre -Dwebshop.ini=<voller Pfad zur webshop.ini> webshop.gui.AuthoringTool

gestartet werden. Wichtig ist hierbei, dass sowohl das WebShop JAR-Archiv als auch der Pfad zur SQL-Klassenbibliothek im Klassenpfad (Umgebungs-Variable *CLASSPATH*) der Java-Umgebung auftauchen. Alternativ kann das Authoring-Tool auch über ein Skript gestartet werden. Passende Skripte für Windows und unix-artige Betriebssysteme liegen dem WebShop bereits bei. In den Skripten sind jeweils die Variablen *WEBSHOP_INI* und *WEBSHOP_JAR* anzupassen. Das Authoring-Tool kann dann mit dem Aufruf *atool.bat* (Windows), bzw. *atool.sh* (Unix und Derivate) gestartet werden. Der Pfad zur SQL-Klassenbibliothek muss allerdings in jedem Fall in der Variable *CLASSPATH* auftauchen. Die Definition von Variablen unter den verschiedenen Betriebssystemen können in den entsprechenden Handbüchern zum Betriebssystem nachgeschlagen werden.

6.3.2 Erläuterung der einzelnen Fenster

In den folgenden Unterkapiteln werden der Aufbau, die Benutzung und die Funktionen der einzelnen Fenster genauer erläutert.

6.3.2.1 Das Hauptfenster

Das Hauptfenster des Authoring-Tools ist - wie der Name bereits vermuten lässt - die zentrale Schnittstelle zwischen Benutzer und Programm. Vom Hauptfenster aus ist es möglich, alle Funktionen zu erreichen, die nötig sind, um den Datenbestand des WebShops zu erstellen und zu verwalten. Das Hauptfenster teilt sich in mehrere Bereiche, die nachfolgend beschrieben sind. Abgebildet ist das Hauptfenster in Abbildung 6.23.

Abbildung 6.23: Das Hauptfenster des Authoring-Tools

Wie bei grafischen Oberflächen üblich befindet sich oben im Fenster eine Menü-Leiste, über welche die einzelnen Funktionen des Programms erreichbar sind. Unter dem Menü befindet sich eine Symbol-Leiste, die die am häufigsten verwendeten Befehle bereit stellt. Zu jedem Symbol gehört ein Eintrag des Programm-Menüs, welcher durch drücken des Symbols aufgerufen wird. Das Programm-Menü unterteilt sich in die Bereiche *'Datei'*, *'Bearbeiten'* , *'Fenster'*, *'Optionen'* und *'?'*. Jeder Bereich stellt verschiedene Menü-Einträge bereit, welche die einzelnen Funktionen des Authoring-Tools aufrufen. Menü-Einträge für nicht ausführbare Funktionen sind ausgegraut, ebenso die entsprechenden Symbole in der Symbol-Leiste. Das Menü *'Datei'* stellt Menü-Einträge zur Verfügung, mit deren Hilfe Artikel und Ordner neu erstellt und entfernt werden können, sowie jeweils einen Menü-Eintrag zum Umbenennen von Ordnern und zum Beenden des Programms. das Menü *'Bearbeiten'* enthält Menü-Einträge zum Bearbeiteten eines Artikels, sowie Menü-Eintrage zum Verwalten der Datenbank-Tabellen für Autoren, Verlage, Artikel-Typen und -Formate, sowie Sprachen. Das Menü *'Fenster'* enthält Menü-Einträge, mit deren Hilfe die verschiedenen Programm-Fenster geöffnet und geschlossen werden können. Ein Menü-Eintrag für ein geöffnetes Fenster ist mit einem kleinen Häkchen versehen. Das Menü *'Optionen'* stellt Menü-Einträge bereit, über welche die Funktionen zur Änderung der Programm-Optionen, zur Festlegung der Einstellungen des WebShops und zur Auswahl eines Layouts für den WebShop aufgerufen werden können. Das Menü *'?'* gibt Informationen zum Authoring-Tool.

In der Mitte des Hauptfensters befindet sich eine geteilte Ansicht für die im WebShop verfügbaren Ordner und den Artikeln, die einem Ordner aktuell zugeordnet sind. In der linken Hälfte der Ansicht werden die verfügbaren Ordner als Baumstruktur dargestellt. Sobald man einen Ordner ausgewählt hat, werden die ihm zugeordneten Artikel in der rechten Hälfte der Ansicht angezeigt. Sollte einem ausgewählten Ordner kein Artikel zugeordnet sein, bleibt die rechte Hälfte der Ansicht leer. Neue Ordner werden über den Menü-Eintrag *'Datei'* → *'Ordner erstellen'*, dem entsprechenden Symbol in der Symbol-Leiste oder über ein Kontext-Menü (Aufruf mittels der rechten Maustaste) erzeugt. Dabei ist zu beachten, dass der neu erzeugte Ordner ein Unterordner des aktuell ausgewählten Ordners ist. Sollte kein Ordner ausgewählt sein, wird der neu erstellte Ordner ein 'Top-Level'-Ordner, also ein Ordner, dem kein weiterer Ordner übergeordnet ist. Das Entfernen von Ordnern verläuft analog, allerdings ist hierbei zu beachten, dass alle Unterordner des zu entfernenden Ordners ebenfalls entfernt werden. Die Zuordnung der Ordner untereinander lässt sich per 'Drag and Drop' ändern. Dazu ist ein Ordner mit der linken Maustaste auszuwählen und bei gedrückter linker Maustaste einfach auf den Ordner zu ziehen, der als neuer übergeordneter Ordner fungieren soll. Das Umbenennen von Ordnern wird über den Menü-Eintrag *'Datei'* → *'Ordner umbenennen'*, dem entsprechenden Symbol in der Menü-Leiste, oder dem Kontext-Menü erreicht. Alternativ kann dies auch erreicht werden, indem ein bereits gewählter Ordner erneut angewählt wird.

Die rechte Hälfte der Ansicht zeigt alle Artikel, die einem gewählten Ordner zugeordnet sind. Neue Artikel können über den Menü-Eintrag *'Datei'* → *'Artikel erstellen'*, dem entsprechenden Symbol in der Symbol-Leiste, oder über ein mit der rechten Maustaste aufrufbares Kontext-Menü erstellt werden. Hierbei ist zu beachten, dass ein neu erstellter Artikel nicht automatisch in den Datenbestand aufgenommen wird. Ein neu erstellter Artikel besitzt zunächst nur einen Namen und ein kleines Standard-Symbol, um ihn zu visualisieren. Damit ein neu erstellter Artikel in den Datenbestand aufgenommen werden kann, ist es notwendig, ihm gewisse Informationen mitzugeben, welche mittels semantisch voneinander getrennter Fenster bearbeitet werden müssen. Diese Fenster sind über die Menü-Einträge im Menü *'Bearbeiten'*, den entsprechenden Symbolen der Symbol-Leiste oder über ein Kontext-Menü zu erreichen. Bei Auswahl einer dieser Menü-Punkte erscheint ein Fenster mit einem Formular, in welches die notwendigen Informationen eingegeben werden können. Die Beschreibung der einzelnen Fenster folgt im Anschluss an dieses Unterkapitel. Erst nach Angabe aller notwendigen Informationen kann ein neu erstellter Artikel mittels des Knopfes *'Speichern'* in einem der Fenster in den Datenbestand aufgenommen und dem ausgewählten Ordner zugeordnet werden.

Die Zuordnung von Artikeln zu Ordnern kann per 'Drag and Drop' verändert werden. Dazu ist ein Artikel auszuwählen und bei gedrückter linker Maustaste auf den Ziel-Ordner zu ziehen. Nach Loslassen der Maustaste wird der entsprechende Artikel dem Ziel-Ordner zugeordnet und die alte Zuordnung entfernt. Sollte beim Loslassen der Maustaste die Steuerungs-Taste der Tastatur niedergehalten worden sein, wird die alte Zuordnung beibehalten. Dadurch ist es möglich, einen Artikel mehreren Ordnern gleichzeitig zuzuordnen. Die Zuordnung zwischen einem Artikel und dem aktuell ausgewählten Ordner kann über den Menü-Eintrag *'Datei'* → *'Artikel entfernen'* aufgehoben werden. Artikel, die keine Zuordnung zu einem Ordner besitzen, können über den Menü-Eintrag *'Bearbeiten'* → *'Entfernte Artikel'* sichtbar gemacht werden. Zudem ist es möglich, mehrere Artikel gleichzeitig auszuwählen. Dazu ist bei der Auswahl die Steuerungs-Taste, bzw. die Hochstell-Taste niederzuhalten. Bereichs-Auswahl von Artikeln wird ebenfalls unterstützt. Allerdings können die Informationen zu einem Artikel nur bearbeitet werden, wenn ein einzelner Artikel ausgewählt ist, um die Eindeutigkeit der Informationen zu wahren.

6.3.2.2 Shop-Einstellungen

Das Fenster mit den Shop-Einstellungen teilt sich in 3 Bereiche auf:

- Generelle Einstellungen
- Internet Einstellungen
- Bezahlungs Einstellungen

In den generellen Einstellungen, die in Abbildung 6.24 zu sehen sind, können allgemeine Einträge vorgenommen werden, wie der Name des Shops und des Besitzer. Weiter kann eine Adresse, eine Telefon-Nummer und eine Fax-Nummer eingetragen werden, unter welcher der Besitzer zu erreichen ist. Mit einem Klick auf den Button neben dem Anzeigefeld vom Root-Verzeichnis wird ein Dateimanager geöffnet. Dort kann das Root-Verzeichnis ausgewählt werden. Dieses Verzeichnis ist das oberste Verzeichnis des Shops. Wird der Shop auf einen anderen Rechner portiert, muss eventuell nur dieser Pfad angepasst werden, die frühere Verzeichnis-Struktur kann dann weiter verwendet werden. Klickt man auf den Button neben dem Anzeigefeld von AGBs, wird ebenfalls ein Dateimanager geöffnet. Hier kann dann eine Datei ausgewählt werden, die die allgemeinen Geschäftsbedingungen beinhaltet.

Abbildung 6.24: Die generellen Shop-Einstellungen

In den Internet Einstellungen kann die Root-Url , also der Teil der Browser-Url die vor die Dokument-Url gesetzt werden muß um eine gültige Adresse zu erhalten, eingetragen werden. Weiter können die Startseite, die Email-Adresse sowie die Adresse des Shop-Logos eintragen werden. Diese Einstellmöglichkeiten sind in Abbildung 6.25 abgebildet.

Abbildung 6.25: Die Internet Shop-Einstellungen

Im Bereich der Bezahlungen können die verschiedenen Bezahlungsarten ausgewählt werden, die der Shop unterstützen soll. Die Auswahl erfolgt durch einen Druck auf das Kästchen neben dem Namen der Zahlungsart.

Abbildung 6.26: Die Bezahlungs Shop-Einstellungen

Mit dem OK-Button werden die Änderungen übernommen, bei Auswahl des Verwerfen-Buttons werden die Änderungen verworfen.

6.3.2.3 Datenbankfenster

Datenbankfenster zeigen die momentanen Inhalte der Datenbank zu einigen Attributen des Dokumentes an und ermöglichen das Hinzufügen von weiteren Werten. Im Authoring-Tool des Webshops gibt es fünf Datenbankfenster, das Autoren-Datenbankfenster, das Format-Datenbankfenster, das Verlag-Datenbankfenster, das Typ-Datenbankfenster und das Sprach-Datenbankfenster. Alle Fenster sind gleich aufgebaut und unterscheiden sich nur inhaltlich. Abgebildet ist ein Datenbankfenster in Abbildung 6.27.

Abbildung 6.27: Ein Datenbankfenster des Authoring-Tools

Im oberen Bereich eines Datenbankfensters stehen die eigentlichen Inhalte. Diese werden ab einer gewissen Anzahl nicht mehr komplett angezeigt und können durch Verschieben des Scrollbalkens sichtbar gemacht werden. Im Autoren-Datenbankfenster besteht die Möglichkeit, die einzlnen Autoren per Drag and Drop in das Autoren-Fenster zu ziehen. Der untere Bereich eines Datenbankfenster dient dem Hinzufügen neuer Inhalte. In das entsprechende Textfeld kann eine beliebige Zeichenfolge eingegeben werden, die durch Bestätigen mittels der Enter-Taste dem Inhaltsfeld und der Datenbank hinzugefügt wird.

6.3.2.4 Details-Fenster

In dem Details-Fenster stehen alle näheren Informationen zu einem Dokument, die nicht bereits in einem der übrigen Fenster dargestellt wurden. Abgebildet ist das Details-Fenster in Abbildung 6.28.

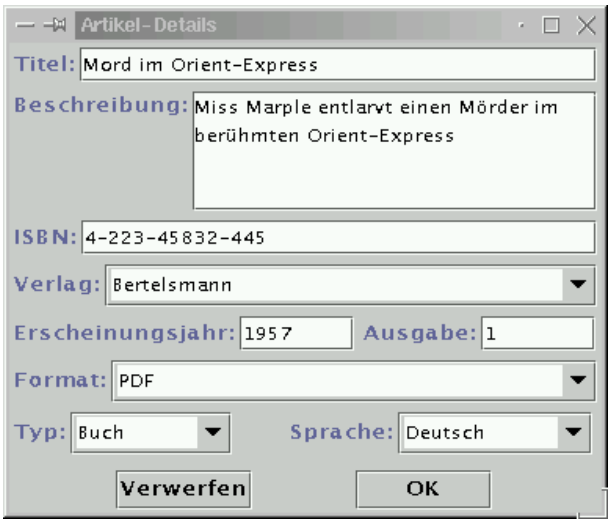

Abbildung 6.28: Das Details-Fenster des Authoring-Tools

Von oben nach unten können hier der Titel, eine Beschreibung, die ISBN, das Erscheinungsjahr und die Ausgabe frei eingegeben werden. Desweiteren kann eine Auswahl des Verlages, des

Formates, des Typs und der Sprache über Combo-Boxen erfolgen. Die Inhalte der Combo-Boxen sind mit den aktuell in der Datenbank enthaltenen Daten identisch. Um einem Dokument also einen neuen Verlag zuzuordnen, muss dieser vorher in der Datenbank bekannt sein. Dies geschieht über das Datenbankfenster für Verlage. Genauso verhält es sich mit den anderen Daten, die über Combo-Boxen ausgewählt werden können. Um diese Daten zu speichern, muss die Schaltfläche 'Speichern' gedrückt werden. Sollen die zuletzt gespeicherten Werte in dem Fenster angezeigt werden, muss die Schaltfläche 'Verwerfen' gedrückt werden.

6.3.2.5 Artikel-Inhalte

Im Fenster für die Artikel-Inhalte (Abbildung 6.29) können die zu einem Dokument gehörenden Urls eingetragen werden.

Im oberen Teil des Fensters ist eine Tabelle mit den Urls des Dokumentes sichtbar. Diese Tabelle hat zwei Spalten. In der ersten wird angezeigt, ob die URL frei oder kostenpflichtig ist. In der zweiten wird die Adresse der URL ab dem in den Shop-Einstellungen gewählten Root-Verzeichnis angezeigt. Mit der Maus und Tastatur können eine oder mehrere Einträge ausgewählt werden. Mit der rechten Maustaste kann ein Kontextmenü geöffnet werden. Hier hat man die Auswahl zwischen:

- Alles Auswaehlen: Alle Einträge werden markiert
- Auswahl aufheben: Alle Markierungen werden aufgehoben
- Nach URL sortieren: Die Einträge werden aufsteigend alphabetisch sortiert
- Nach Frei sortieren: Alle als Frei markierten Einträge werden zuerst angezeigt, dann erst die kostenpflichtigen
- Markierte freisetzen: Die markierten Einträge werden auf Frei gesetzt
- Markierte sperren: Die gewählten Einträge werden als kostenpflichtig markiert
- Markierte loeschen: Die ausgewählten Einträge werden gelöscht

Drückt man den Button mit der Aufschrift "Urls auswaehlen" unter der Tabelle, wird ein Dateimanager geöffnet. Hier können die zum Dokument gehörenden Dateien ausgewählt werden. Im unteren Teil des Fensters können die Adressen für das Inhaltsverzeichnis, die Homepage, das Symbol für das Dokument und für eine Leseprobe ausgewählt werden. Auch dies geschieht wieder über den jeweiligen Button. Dieser öffnet dann einen Dateimanager, indem die Adresse gewählt werden kann.

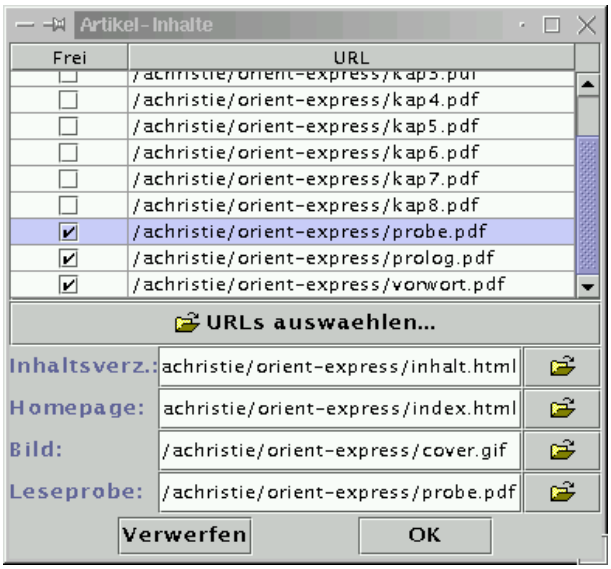

Abbildung 6.29: Auswahl der Artikel-Inhalte

6.3.2.6 Autoren-Fenster

Das Autoren-Fenster stellt die zu einem Dokument gehörenden Autoren bzw. Herausgeber dar. Außerdem ist es über dieses Fenster möglich, dem Dokument neue Autoren bzw. Herausgeber hinzuzufügen. Abgebildet ist das Autoren-Fenster in Abbildung 6.30.

Abbildung 6.30: Das Autoren-Fenster des Authoring-Tools

Aufgeteilt ist das Fenster in drei Spalten, der Namensspalte, der Herausgeberspalte und der Autorenspalte. Die Namensspalte enthält die Namen der Autoren bzw. Herausgeber. Um einen Namen als Autoren bzw. Herausgeber des Dokumentes zu markieren, muss in der gleichen Zeile in der Autoren- bzw. Herausgeberspalte die Checkbox mit einem Haken versehen werden. Einem neuen Dokument sind zu Anfang noch keine Namen in dem Autoren-Fenster zugeordnet. Mittels Drag And Drop können neue Namen aus dem Autoren-Datenbankfenster dem Autoren-Fenster hinzugefügt werden, wobei diese initial weder Autor noch Herausgeber sind. Diese Zuordnung muss explizit erfolgen. Sind mindestens zwei Zeilen in dem Autoren-Fenster zu einem Dokument vorhanden, können durch Anklicken mittels der rechten Maustaste einzelne Zeilen gelöscht werden. Es muss jedoch immer eine Zeile stehen bleiben. Die Reihenfolge der Zeilen im Autoren-Fenster von oben nach unten gibt die Reihenfolge der Nennungen der Autoren bzw. Herausgeber zu dem Dokument auf der Kundenseite vor. Per Drag And Drop kann die Reihenfolge verändert

werden. Um die eingetragenen Daten in die Datenbank zu schreiben, muss auf die Schaltfläche 'Speichern' gedrückt werden. Damit die zuletzt gespeicherten Werte im Fenster erscheinen, muss auf die Schaltfläche 'Verwerfen' gedrückt werden.

6.3.2.7 Angebote-Fenster

Das Angebote-Fenster enthält alle zu einem Dokument jemals erstellten Angebote und kennzeichnet ein Angebot als zur Zeit gültig. Es ist über dieses Fenster möglich, dem Dokument neue Angebote hinzuzufügen. Abgebildet ist das Angebote-Fenster in Abbildung 6.31.

Abbildung 6.31: Das Angebote-Fenster des Authoring-Tools

Es existieren zwei Bereiche, das obere Panel ist der Darstellungsbereich der bisher eingetragenen Angebote und das untere Panel ist der Angebots-Erstellungsbereich. Im oberen Bereich stehen die Angebote zu dem Dokument in DM oder Euro und einer Markierung, ob das jeweilige Angebot zur Zeit Gültigkeit besitzt. Diese Gültigkeit wird durch einen Haken in der links neben dem Angebot stehenden Checkbox ausgedrückt. Es kann nur ein Angebot zur Zeit gültig sein. Um ein neues Angebot dem Dokument hinzuzufügen, muss ein Betrag in das Textfeld mit der Bezeichnung 'Neues Angebot' eingetragen werden. Diesem Betrag kann nun durch Auswahl des DEM-Buttons bzw. des EUR-Buttons eine entsprechende Währung zugeordnet werden. Wenn alle Daten korrekt eingegeben wurden, kann das neue Angebot durch Drücken der Enter-Taste in die obere Liste übertragen werden. Soll das neue Angebot auch Gültigkeit besitzen, muss dieses durch Markieren der entsprechenden Checkbox explizit geschehen. Wenn die Daten in die Datenbank geschrieben werden sollen, muss die Schaltfläche 'Speichern' gedrückt werden. Sollen die zuletzt gespeicherten Daten in dem Fenster angezeigt werden, so muss die Schaltfläche 'Verwerfen' gedrückt werden. Es besteht keine Möglichkeit, Angebote zu löschen.

6.3.2.8 Der Dateimanager

Um dem Benutzer die Auswahl von Verzeichnissen oder Dateien zu erleichtern, öffnet sich beim Druck bestimmter Buttons in einigen Fenstern der Dateimanager. Die Oberfläche des Dateimanagers teilt sich in 2 wesentliche Bereiche auf, wie in Abbildung 6.32 zu sehen ist.

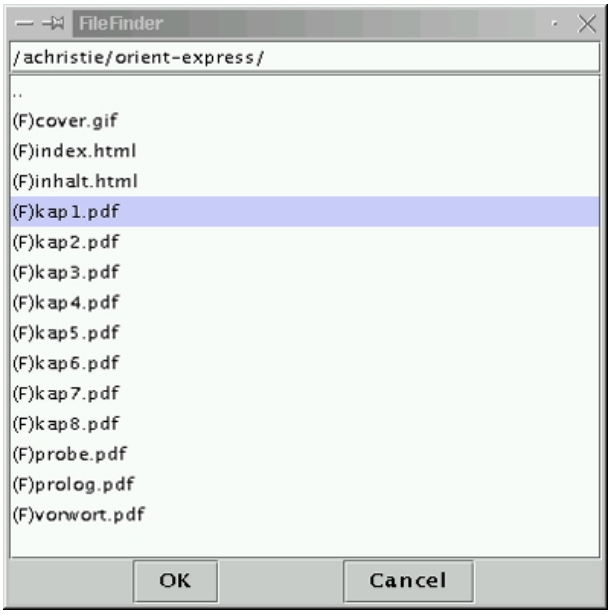

Abbildung 6.32: Der Dateimanager

Oben befindet sich ein Adressfeld indem das aktuelle Verzeichnis ab dem in den Shop-Einstellungen gewählten Root-Verzeichnis dargestellt wird. Durch Eingabe einer Adresse kann hier direkt zu einem Verzeichnis gesprungen werden.
Der größte Teil des Fensters wird durch den eigentlichen Datei-Browser bestimmt. Vor dem Addressnamen steht dabei ein (D) für ein Verzeichnis und ein (F) für eine Datei. Durch Doppel-Klick auf ein Verzeichnis kann in dieses gewechselt werden. Durch Wahl von ".." kann man in die nächst höhere Verzeichnis-Ebene wechseln. Dies jedoch auch nur solange, bis das gewählte Root-Verzeichnis erreicht wurde.
Abhängig davon, von welchem Ort aus der Dateimanager aufgerufen wurde, können eine oder mehrere Dateien oder Verzeichnisse ausgewählt werden.

Aufruf von	Mögliche Auswahl
Shop-Einstellungen	
Root-Verzeichnis	1 Verzeichnis
AGBs	1 Datei
Artikel-Inhalte	
URLs auswaehlen	1 oder mehrere Dateien
Inhaltsverzeichnis	1 Datei
Homepage	1 Datei
Bild	1 Datei
Leseprobe	1 Datei

Bei Druck auf OK werden die gewählten Adressen übernommen. Beim Druck auf Cancel wird der Dateimanager ohne Änderungen geschlossen.

6.3.3 Vorgehensweise bei den wichtigsten Arbeitsschritten

Im Folgenden wird in Form einer To-Do-Anleitung die Vorgehensweise bei den wichtigsten Arbeitsabläufen geschildert. Es wird jedoch nicht die genaue Bedienung der einzelnen Fenster erklärt. Für diese Beschreibung sei der Benutzer auf die Kapitel 6.3.2.1 bis 6.3.2.8 verwiesen.

- Shop-Einstellungen ändern

  1. Aufruf der Shop-Einstellungen
     Rufen Sie über das Optionen-Menü im Hauptfenster die Shop-Einstellungen auf.
  2. Auswahl des Bereiches
     Wählen Sie den Bereich aus, in dem Sie Änderungen vornehmen möchten (Generell, Internet oder Bezahlung).
  3. Eintragen der Informationen
     Nehmen Sie die gewünschten Änderungen vor. Wenn Sie keine weiteren Änderungen mehr vornehmen möchten, drücken Sie auf *OK* zum speichern oder auf *Verwerfen* zum verwerfen der Änderungen.

- Neues Dokument anlegen

  1. Vorbereiten
     Gehen Sie in den Ordner, in dem das neue Dokument erstellt werden soll. Wählen Sie dann über den Button oder das Kontextmenü (rechte Maustaste) die Funktion *Artikel erstellen* aus. Ein neues Artikelsymbol wird angezeigt. Dieser Artikel enthält aber noch keine Informationen.
  2. Eingabe der Informationen
     Wählen Sie jetzt wie bei *Dokument ändern* einen Bereich aus, indem Sie zuerst Informationen eintragen wollen. Wenn Sie nach dem Eintragen auf den Speichern-Button klicken, werden Sie eventuell vom Programm darauf aufmerksam gemacht, daß noch Informationen fehlen. Gehen Sie jetzt in den nächsten Bereich und füllen Sie auch dort die Eingabefelder. Erst wenn alle Felder in allen Bereichen korrekt eingetragen wurden, wird das Programm versuchen das neue Dokument an die Datenbank weiterzuleiten. Nach erfolgreichem Eintrag in die Datenbank bekommen Sie eine Bestätigung vom Programm.

- Dokument ändern

  1. Auswahl
     Wählen Sie das zu ändernde Dokument aus. Sie können jetzt über die auswählbar gewordenen Buttons oder das Kontextmenü den zu ändernden Bereich der Dokument-Informationen wählen.

2. Ändern der Informationen
   Nehmen Sie in dem neu geöffneten Fenster die Änderungen vor. Drücken Sie auf *OK* zum speichern oder auf *Verwerfen* zum verwerfen der Änderungen. Wollen Sie weiter Informationen aus anderen Bereichen ändern, gehen Sie diese Arbeitsschritte nochmal durch.

- Dokument löschen

  1. Auswahl
     Gehen Sie in das Verzeichnis mit dem gewünschten Dokument und wählen Sie es aus.
  2. Löschen
     Sie können das Dokument dann über das Kontextmenü (rechte Maustaste) oder den dafür bestimmten Button löschen.

- Dokument verschieben

  1. Vorbereiten
     Sorgen Sie dafür, daß das Ziel-Verzeichnis im linken Fenster des Hauptfensters sichtbar ist. Gehen Sie dann zum gewünschten Dokument.
  2. Verschieben
     Wählen Sie das Dokument aus und halten Sie die linke Maustaste gedrückt, während sie das Dokument auf das Ziel-Verzeichnis ziehen. Wenn Sie die Maustaste lösen, wird das Dokument verschoben.

Kapitel 7

DDS - Web - Komponente

7.1 Einleitung

Die Aufgabe der DDS-Web-Gruppe war es, mit Hilfe von Java (in der Version 1.2) und der auf Java aufbauenden Technologie der Servlets einen Internetshop aufzubauen, der mit den weiteren Komponenten des DDS-Systems (DDS-Datenbank, DDS-Access, DDS-Payment und DDS-Authoring) zusammenarbeitet und die Kommunikationsschnittstelle zwischen dem Kunden und dem DDS-System in Form von dynamischen Web-Seiten realisiert.

Der folgende Teil des Berichts befasst sich mit der Erstellung der DDS-Web-Komponente, in dem die folgenden Unterpunkte besprochen werden:

- Der Entwurf der DDS-Web-Komponente.
- Die Architektur der DDS-Web-Komponente.

Als zweiter Teil folgt das Benutzungshandbuch mit den Screenshots zu den einzelnen Oberflächenlayouts und den enthaltenen Teil-Fenstern. Dort wird erklärt, wie der Betreiber und der Benutzer des DDS - Systems vorgehen müssen, damit sie den iShop möglichst effektiv nutzen können.

Am Ende dieses Kapitels werden die Klassenstrukturen grafisch dargestellt.

7.2 Entwurf und Ipmlementierung

Dieser Abschnitt beschreibt die Vorgehensweise der Gruppe bei der Implementierung. Dazu gehört der Meilensteinplan, der Entwurf und die Architektur der Servlets.

7.2.1 Der Meilensteinplan

Da der Gruppe die Erfahrungen in der Servlet-Programmierung fehlte, ging die Gruppe direkt von der Experimentier- und Test-Phase (also wie man mit Servlets arbeitet und was sie bewirken) in die Implementierungs-Phase über. Deswegen lag der erste Meilenstein in der Mitte des Monats August. Bis zu diesem Zeitraum sollte dann die Implementierung weitestgehend abgeschlossen sein. Nach der Implementierung erfolgte die Integration mit den weiteren DDS-System-Komponenten, die von den anderen Teilgruppen realisiert wurden. Die Dokumentation sollte im Anschluss an die Integrationsphase angegangen werden und möglichst bis Ende September fertiggestellt werden.

7.2.2 Der Entwurf und die Implementierung

Zur Realisierung des WebShops wurde Java 1.2 von Sun Microsystems gewählt. Dabei wurde das Java Development Kit (JDK) 1.2.2, sowie das Java Server Development Kit (JSDK) 2.0, Apache in der Version 1.3.12 und der ApacheJServer in der Version 1.1 verwendet.

Aufgrund der in der Oberlächengestaltung getroffen Designentscheidungen wird bei den Browsern Javascript und Frame Unterstützung vorausgesetzt. Da die Sessionidentifizierung über Cookies geht, müssen diese im Browser eingeschaltet sein.

7.2.2.1 Die Oberfläche

Ausgehend von der benötigten Funktionalität des WebShop's wurden zunächst von jedem der Gruppenmitglieder Design-Überlegungen angestellt, die dann in drei, zunächst in statischem HTML programierten, Demoversionen der GUI festgehalten wurden. Da allen drei Versionen gemeinsam war, das ein Frameset benutzt wurde, in dem die Hauptaktivität jeweils in einem Frame stattfand, wobei diese Hauptaktivitäten, aufgrund der Aufgabenstellung, ziemlich gleichartig aufgebaut war, haben wir uns entschieden, diese drei Gui-Versionen auch in dem iShop zu implementieren.

7.2.2.2 Die Entwicklung der Servlets

Da aufgrund der Designentscheidung verschiedene Frames mit Inhalten zu versorgen waren, bot es an, sich an diese Inhalte jeweils durch spezielle Servlets zu erstellen. Dadurch konnte der Sourcecode der einzelnen Servlets relativ kurz und übersichtlich gehalten werden. Dadurch mussten aber auch mehrere Servlets implementiert werden, die in wesendlichen Teilen übereinstimmen. Diese Servlets mussten alle :

1. Die Möglichkeit haben auf die kundenspezifischen Daten, die in der laufenden Session erzeugt wurden zuzugreifen, und diese auch wieder so abzulegen das andere Servlets die in dieser Session noch aufgerufen werden wieder darauf zugreifen können.

2. Sämtliche für die Datenverarbeitung nötigen Parameter vor dem Begin der Datenverarbeitung aus dem HttpRequest auslesen, um sicherzustellen das die Verarbeitung nicht wegen fehlender Daten scheitert.

3. Die Ausgabe der erzeugten HTML-Seite erst nach der kompletten Verarbeitung erzeugen, wodurch verhindert wird das durch einen Fehler in der Verarbeitung eine fehlerhafte HTML-Seite ausgegeben wird.

Um dieses Verhalten für alle Servlets sicherzustellen und es doch nicht jedesmal neu ausprogramieren zu müssen, wurde die abstrakte Klasse *webshop.servlet.***Shoplet** entworfen, die den Zugriff auf die kundenspezifischen Daten ermöglicht und die Ausgabe der erzeugten HTML-Seite übernimmt. In den abgeleiteten Klassen, welche dann die Servlets sind, müssen zwei Methoden implementiert werden:

boolean **getAllParameters():** In dieser Methode werden die für dieses Servlet benötigten Parameter aus dem HttpServletRequest ausgelesen und in einer Hashtable für den späteren Zugriff gespeichert. Wenn alle nötigen Parameter im HttpServletRequest gefunden werden wird *true* zurückgeliefert.

ShopPage **doIt(**ShopletData data**):** In dieser Methode wird die eigentliche Arbeit geleistet. Es wird die Datenverarbeitung durchgeführt und als Resultat eine HTML-Seite, in Form einer *webshop.servlet.* **ShopPage**, erzeugt und zurückgeliefert.

Für die Erzeugung der Html-Ausgabe wurde eine Klassenstruktur entwickelt, bei der, während der Erzeugung der Ausgabe nur noch wenig auf das Design geachtet werden muss, da dieses bei der Ausgabe durch die Basisklassen *webshop.servlet.FramePage*, für die Rahmenframes, und *webshop.servlet.ContentPage*, für das Hauptframe erledigt wird. Die Methode doIt() der Servlets liefert als Resultat die Instanz einer von diesen Basisklassen abgeleiteten Klasse zurück, welche dann ausgegeben wird.

Um die Handhabung der verschiedenen HTML-Elemente zu erleichtern, wurden einige Interfaces und Klassen entworfen, von denen einige Beispiele hier etwas erläutert werden :

Das Interface *FormContent* definiert die Methode *toHtml()*, eine Instanz einer dieses Interface implementierenden Klasse kann direkt eine HTML-Ausgabe des Objektes erzeugen.

Die Klasse *FormInput* repräsentiert ein HTML <INPUT> Element. Sie imlementiert das Interface *FormContent*. Sie erleichtert die Handhabung der vielen verschiedenen <INPUT> Elemente in den Formularen, die auf den Html-Seiten benötigt werden.

Das Interface *FormSelector* erweitert das Interface FormContent um die Methoden die fuer die Bereistellung eines HTML <SELECT> Elementes implementiert werden müssen.

Die Klasse *RangeSelector* repräsentiert ein HTML <SELECT> Element. Sie imlementiert das Interface *FormSelector*. Sie ermöglicht die einfache Erzeugung vom <SELECT> Elementen mit numerischem Auswahbereich.

Die Klassen *BankAccountFormContent* **und** *CreditCardFormContent* erweitern die Basisklassen *BankAccount* und *CreditCard* um die Methode *toHtml()* (das Interface FormContent), welche die Formularelemente zur Aufnahme der jeweiligen Daten erzeugt, und um einen Konstruktor, welcher den *HttpServletRequest* als Argument erhält und daraus die Eingaben ausliest.

7.2.3 Die Architektur des Webshop's

Der WebShop besteht aus insgesammt 11 Servlets (Abb. 7.1) durch welche die in den Frames benötigeten Funktionalitäten bereitgestellt werden. Die Servlets sind nach ihrer Funktion bzw. ihrer erzeugeten Ausgabe benannt. Die Klassen, welche die Html-Ausgabe erzeugen sind wie die dazugehörigen Servlets benannt: Die Servlets heißen meist *Viewer und die entsprechenden Html-Code generierenden Klassen entsprechend *Page (Abb. 7.2).

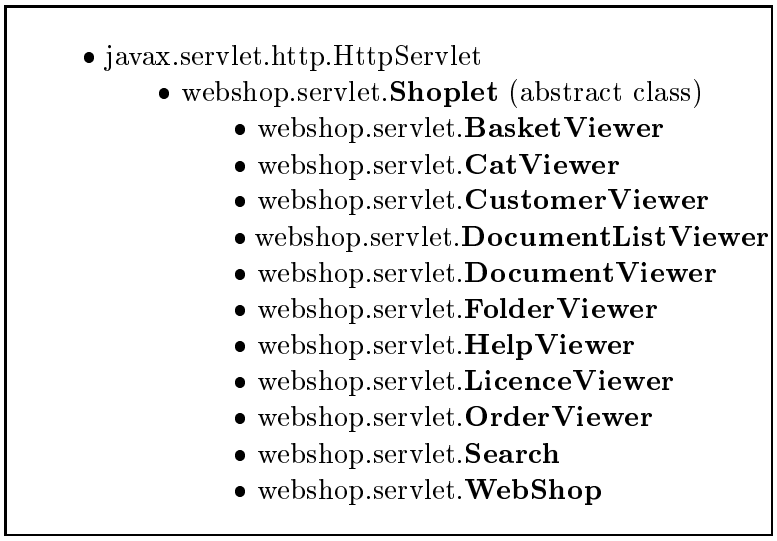

Abbildung 7.1: Die Klassenstruktur der Servlets

Das Servlet **Webshop**

Als Einstiegspunkt in den WebShop, sowie für die Html-Seiten deren Erzeugung keine weitere Funktionalität benötigt, dient das Servlet *webshop.servlet.***WebShop**. Es liefert, wenn es ohne einen definierten Parameter *webshop.servlet.***DOIT** im *HttpServletRequest*, angefordert wird das Frameset für den WebShop, durch welches dann die weiteren Frameinhalte von den Servlets angefordert werden. Das Servlet WebShop erzeugt die Html-Seiten:

- **FramePage:** das Frameset.
- **LogoPage:** das Frame mit dem ShopLogo (oben links).
- **NavigationPage:** in Layout 1 die obere Navigationsleiste.
- **InfoPage:** die Seiten die als Start- und Info-seiten gezeigt werden, wenn diese nicht in der Datenbank definiert sind.
- **HeadPage:** in Layout 3 das Frame rechts oben.
- **ErrorPage:** eine Fehlermeldung welche den Nutzer über aufgetretene Fehler, oder seine abgelaufene HttpSession informiert.

- webshop.servlet.**ShopPage** (abstract class)
  - webshop.servlet.**FramePage**
  - webshop.servlet.**ShopContentPage** (abstract class)
    - webshop.servlet.**BasketPage**
    - webshop.servlet.**CustomerPage**
    - webshop.servlet.**DocumentListPage**
    - webshop.servlet.**DocumentPage**
    - webshop.servlet.**ErrorPage**
    - webshop.servlet.**HelpPage**
    - webshop.servlet.**InfoPage**
    - webshop.servlet.**LicencePage**
    - webshop.servlet.**LoginPage**
    - webshop.servlet.**OrderPage**
    - webshop.servlet.**SearchPage**
  - webshop.servlet.**ShopFramePage** (abstract class)
    - webshop.servlet.**CatPage**
    - webshop.servlet.**EmptyFramePage**
    - webshop.servlet.**FolderPage**
    - webshop.servlet.**LogoPage**
    - webshop.servlet.**NavigationPage**
    - webshop.servlet.**HeadPage**

Abbildung 7.2: Die Klassenstruktur der ShopPages

Das Servlet **BasketViewer** dient der Anzeige und Änderung der Mengen und Zeiten des Warenkorbinhalts. Über den Warenkorb kann für eine Bestellung der OrderViewer aufgerufen werden. Das Servlet BasketViewer erzeugt die Html-Seiten:

- **BasketPage:** für die Darstellung des Warenkorbs.
- **OrderPage:** zur Eingabe der Bestellung.
- **LoginPage:** wenn eine Bestellung gemacht wird bevor der Kunde sich angemeldet hat.

Das Servlet **CatViewer** generiert die Anzeige der Kategorien, die im Layout 2 oben rechts angezeigt werden. Das Servlet CatViewer erzeugt die Html-Seite:

- **CatPage:** in Layout 2 die Anzeige der Kategorien.

Das Servlet **CustomerViewer** deckt die Funktionen ab, welche für die Benutzerverwaltung benötigt werden. Das sind Kunden anlegen bzw. Kundendaten bearbeiten, Kontoauszug anzeigen und Kundensitzung abmelden. Das Servlet CustomerViewer erzeugt die Html-Seiten:

- **LoginPage:** wenn dieses Servlet aufgerufen wird, bevor anderswo ein Login benötigt wurde.
- **CustomerPage:** zur Darstellung der Kundendaten, des Kontoauszuges oder der Lizenzen.
- **OrderPage:** zum Aufladen des Kundenkontos.

Das Servlet **DocumentListViewer** zeigt den Inhalt einer Kategorie an. Das Servlet DocumentList-Viewer erzeugt die Html-Seite:

- **DocumentListPage:** für die Darstellung der Liste der in einer Kategorie enthaltenen Dokumente.

Das Servlet **DocumentViewer** zeigt alle zu einen Dokument verfügbaren Informationen an.Über den DocumentViewer kann für eine Bestellung der OrderViewer aufgerufen werden. Das Servlet DocumentViewer erzeugt die Html-Seiten:

- **DocumentPage:** zur Anzeige der Dokumentinformationen.
- **LoginPage:** wenn eine Bestellung gemacht wird bevor der Kunde sich angemeldet hat.
- **BasketPage:** zur Aufnahme der Bestelldaten für das Dokument.

Das Servlet **FolderViewer** erzeugt in den Layout's 1 und 3 die Übersicht der Kategorien im linken Frame. In Layout 3 werden auch die Navigationslinks im linken Frame erzeugt. Das Servlet FolderViewer erzeugt die Html-Seite:

- **FolderPage:** zur Generierung des linken Frames mit Navigation (nur Layout 3) und Kategoriebaum (Layout 1 und 3).

Das Servlet **HelpViewer** generiert die Online-Hilfe. Das Servlet HelpViewer erzeugt die Html-Seite:

- **HelpPage:** die Darstellung der Online-Hilfe.

Das Servlet **LicenceViewer** generiert die Anzeige der Lizenzen eines Kunden. Das Servlet Licence-Viewer erzeugt die Html-Seiten:

- **LicencePage:** zur Anzeige der Lizenzen eines Kunden.
- **LoginPage:** wenn dieses Servlet aufgerufen wird, bevor anderswo ein Login gemacht wurde.

Das Servlet **OrderViewer** bearbeitet die Durchführung der Bestellung von Lizenzen, sowie das Aufladen des Kundenkontos. Das Servlet OrderViewer erzeugt die Html-Seiten:

- **OrderPage:** zur Verarbeitung der Bestellung.
- **LoginPage:** wenn dieses Servlet aufgerufen wird, bevor anderswo ein Login gemacht wurde.
- **CustomerPage:** zum Einrichten eines neuen Kunden.

Das Servlet **Search** ermöglicht die Suche auf der Dokumentendatenbank. Das Servlet Search erzeugt die Html-Seite:

- **SearchPage:** zur Eingabe der Suchdaten sowie der Darstellung der Suchergebnisse.

Eine Übersicht über die weiteren implementierten Interfaces und Klassen zeigen die Abbildungen 7.20 bis 7.22.

7.3 Benutzerhandbuch

Dieses Benutzerhandbuch teilt sich in zwei Abschnitte auf. Zum einen werden die wichtigsten Erklärungen und Erläuterungen für den Betreiber des DDS-Systems aufgezählt, zum anderen die Beschreibungen, um mit dem Programm zu arbeiten, was für den späteren Benutzer des Programms von Bedeutung ist.

Der Betreiber benötigt vor allem Informationen zur Installation und Wartung des WebShops. Ausserdem muss der Betreiber wissen, welche Voraussetzungen sein System erfüllen sollte, damit dieser Teilbereich des DDS-System einwandfrei funktioniert.

Für den Benutzer ist vor allem die Nutzung des WebShop wichtig. Deswegen werden für den Benutzer die verschiedenen Interaktionsmöglichkeiten erläutert. Dabei stellt der WebShop drei verschiedene Oberflächen zur Verfügung, die allerdings nur vom Betreiber des Online-Shops gewählt werden können. Trotzdem wird in diesem Benutzerhandbuch auf alle drei Oberflächen eingegangen, und es werden die Unterschiede dargestellt.

In allen drei Layouts wurde die Oberfläche jeweils in vier Teile aufgeteilt. Dabei befindet sich in allen drei Oberflächen in der linken oberen Ecke das Logo des Shops. Ausserdem werden die wichtigsten Anzeigen bei allen drei Oberflächen im rechten unteren Teil des Fensters dargestellt. Es befindet sich bei allen drei Layouts im rechten unteren Teil der Oberfläche der Hauptframe, der den grössten Teil der Oberfläche einnimmt. In ihm können Sie alle weiteren Aktionen durchfüren. Im Gegensatz zu den anderen Frames wechselt hier je nach Aktion der Inhalt, sonst wird in den anderen Frames nur gelegentlich eine Aktualisierung durchgeführt.

Alle drei Layouts laden als Startseite, im rechten unteren Teil des Fensters, die vom Betreiber angegebene Seite ein. Wird vom Betreiber allerdings keine Seite angegeben, lädt der Shop eine selbsterstellte Seite ein, mit den wichtigsten Informationen über den Shop, ein. Diese Seite ist identisch mit der ShopInfo-Seite, sofern der Betreiber hier auch keine eigene Seite zur Verfügung gestellt hat.

7.3.1 Erklärungen für den Betreiber

Um diesen Teil des DDS-Systems zum laufen zu bringen, müssen natürlich die anderen Teile des DDS-Systems installiert werden.

7.3.1.1 Installation und Inbetriebnahme des WebShops durch den Betreiber

Die genauen Installationsvoraussetzungen und -anweisungen können Sie jederzeit am Anfang dieses Berichts nachschlagen. Im Kapitel 2 finden Sie die genaue Anleitung zu diesen Vorgängen, damit Sie später als Betreiber keine Schwierigkeiten mit dem Einrichten des DDS-Systems haben.

7.3.2 Handbuch für den Benutzer

Hier folgen ein paar kurze Erläuterungen zu den Begriffen die in diesem Benutzerhandbuch verwendet werden, damit die Beschreibungen der Layouts nicht zu abstrakt werden (Bsp.: im linken unteren Teil des Hauptfenster). Die komplette WebShop Oberfläche wird als Oberfläche bezeichnet, diese Oberfläche wird in vier Frames, das heisst Teile, aufgeteilt. Wenn im Text zum Beispiel steht: „im rechten unteren Frame passiert folgendes“, ist damit gemeint, daß der rechte untere Teil der Oberfläche gemeint ist. Als letzter Begriff soll noch der Hauptframe eingeführt werden, dieser befindet sich bei allen drei Layouts auf der rechten Seite im unteren Teil, und nimmt den größten Teil der Oberfläche ein.

7.3.2.1 Installationshinweise für den Benutzer

Für Sie als Benutzer ergibt sich kein Installationsaufwand, denn sofern Sie einen Computer mit einem Betriebssystem und einem Standard-Browser mit eingeschaltetem JavaScript, Frame Unterstützung und dem Akzeptieren von Cookies zur Verfügung haben, brauchen Sie nur noch einen Provider der Ihnen den Zugang zum Internet zur Verfügung stellt. Sind diese Bedingungen alle erfüllt, können Sie mit Hilfe des Browsers die jeweilige URL des WebShops eingeben, und Sie landen auf der Startseite des Online-Shops.

7.3.2.2 Die verschiedenen Layouts

Von der DDS-Web-Gruppe wurde entschieden, drei verschiedene Layouts zu erstellen, damit dem Betreiber des DDS-Systems zumindest eine kleine Auswahl verschiedener grafischer Shop-darstellungen zur Verfügung steht. An dieser Stelle finden Sie jetzt die näheren Unterschiede der einzelnen Layouts, die Sie als Benutzer beachten müssen.

Layout 1:

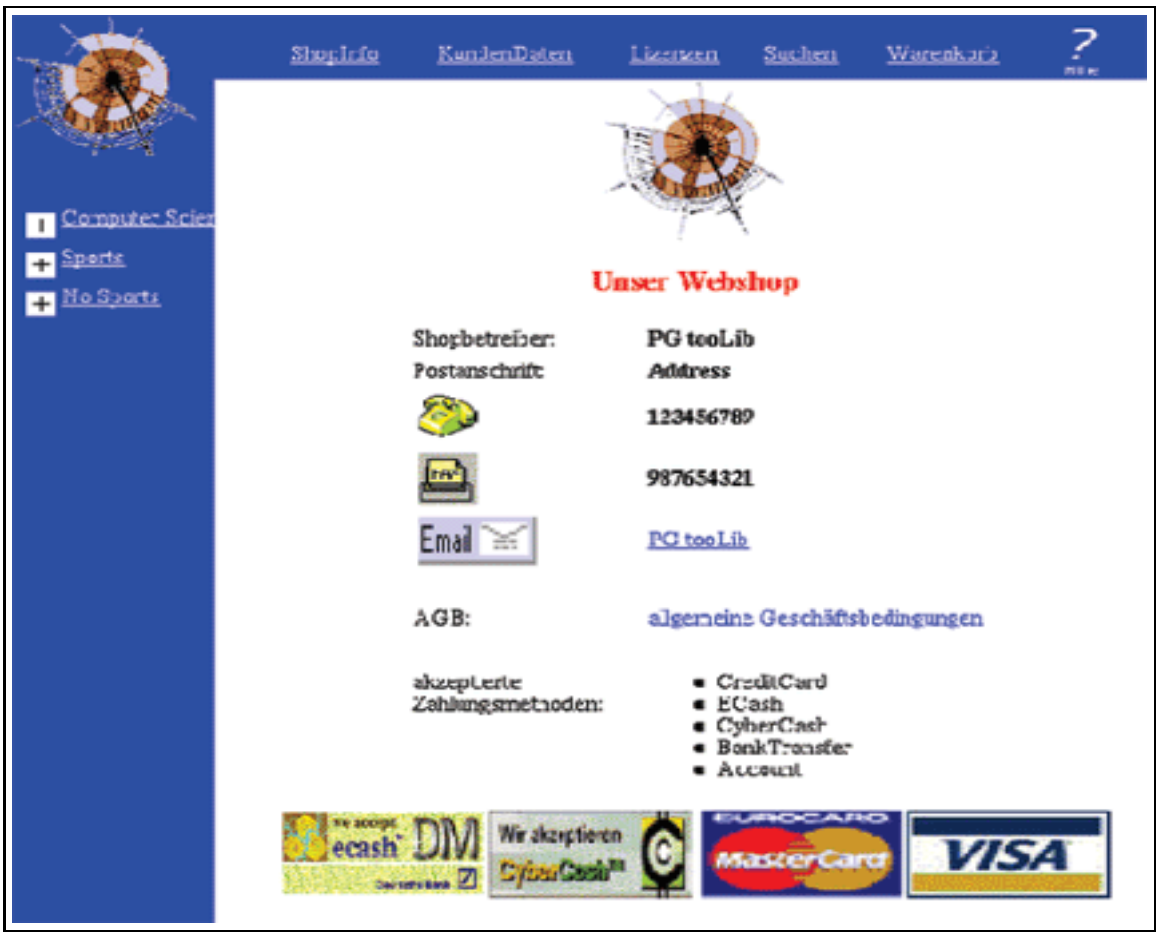

Abbildung 7.3: Das Layout 1

Das Layout 1 (Abb. 7.3) teilt sich ebenso wie die anderen beiden Layouts in vier Frames auf. Oben links befindet sich das Logo des Shops. In diesem Frame gibt es sonst keine weiteren Interaktionsmöglichkeiten mehr.

Im linken unteren Frame befindet sich die Navigationsleiste, um durch die einzelenen Dokumentkategorien (Abb. 7.4), die vom aktuellen Online-Shop angeboten werden, zu navigieren. Der Aufbau der Dokumentkategorien ist mittels einer Verzeichnisbaumstruktur realisiert. Zu Beginn bekommen Sie nur die Hauptkategorien angezeigt, diese können Sie dann aber durch Anklicken weiter aufklappen. Mit den Hauptkategorien sind die Dokumentverzeichnisse gemeint, denen kein weiteres Verzeichnis über gestellt ist. So ergibt sich dann eine komplette Baumstruktur durch die verschiedenen Kategorien. Klicken Sie auf ein Plus-Symbol wird der Baum weiter aufgeklappt, klicken Sie auf ein Minus-Symbol wird der Baum wieder eingeklappt und wenn Sie direkt auf den Namen einer Kategorie klicken, dann wird, falls dies noch nicht geschehen ist, der Baum weiter aufgeklappt und im Hauptframe erscheint die Dokumentanzeige, die alle Dokumente dieser Kategorie auflistet.

Im rechten oberen Frame befindet sich ebenfalls eine Navigationsleiste (Abb. 7.5), die sich aber

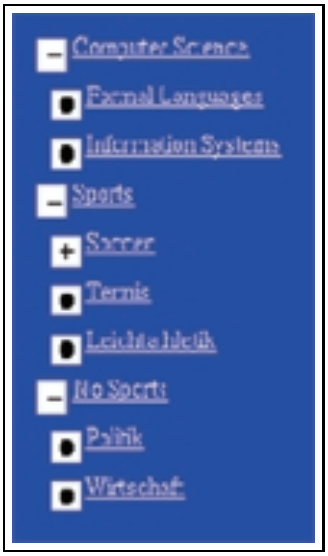

Abbildung 7.4: Die Navigation durch die Dokumentkategorien im Layout 1

Abbildung 7.5: Die Navigationsleiste von Layout 1

nicht auf den Dokumentbestand bezieht, sondern auf die Navigation durch den Web Shop. Jeder der in der Navigationsleiste vertretenen Menüpunkte zeigt im Hauptframe den entsprechenden Inhalt an. Die Navigationsleiste stellt folgende Menüpunkte zur Verfügung:

ShopInfo Mit diesem Menüpunkt wird die Shop Informations Seite aufgerufen. Dies ist entweder eine Seite, die vom Betreiber erstellt wurde und seinen Vorstellungen entspricht, oder aber es wird eine Seite eingeladen, die von den Servlets erstellt wurde und die nötigsten Informationen über den Shop enthält.

KundenDaten Wenn der Benutzer auf den Menüpunkt „KundenDaten“klickt, erscheint im Hauptframe ein Menü das folgende Menüpunkte zur Verfügung stellt.

persönliche Daten bearbeiten Hier gelangen Sie zu einem Fenster in dem Sie sich gegebenenfalls anmelden oder Ihre Nutzerdaten bearbeiten und verändern können. Dazu gehört auch der Wechsel Ihres Passwortes.

Kontoauszug Hier können Sie sich Ihren Kontoauszug anzeigen lassen, das heisst die Buchungen die innerhalb des letzten Monats vorgenommen wurden, werden Ihnen untereinander aufgelistet. Diese Liste läst sich aber durch die entsprechenden Buttons auch weiter zurückverfolgen.

Kundenkonto aufladen Um Ihr Kundenkonto aufzuladen müssen Sie auf diesen Link klicken.

Benutzer Logout Sofern Sie sich ausloggen wollen, um eventuell den Benutzer zu wechseln oder den Shop zu verlassen, müssen Sie auf diesen Link klicken. Der Warenkorb wird hierbei allerdings nicht gelöscht, damit Sie falls Sie Ihre Identität innerhalb des iShops ändern wollen nicht alle Dokumente erneut zusammen suchen müssen. Dies kann der Fall sein, wenn Sie in Warenkorb abgelegte Bücher unter einem anderen Benutzerlogin kaufen wollen.

Lizenzen Es gibt zwei Arten von Lizenzen: die gültigen Lizenzen und die komplette Liste aller Lizenzen, das heisst zur Liste der gültigen Lizenzen kommen noch die Lizenzen hinzu, die noch nicht freigeschaltet sind. Unter diesem Menüpunkt werden erst die gültigen Lizenzen angezeigt, die kompletten Lizenzen werden angezeigt, wenn man auf den Button für „alle Lizenzen“klickt.

Suchen Ein Mausklick auf diesen Link führt Sie zur Suchseite, wo Sie die Möglichkeit haben, nach einem oder mehreren Suchbegriffen im Dokumentbestand zu suchen. Insgesamt können sie in sechs Kategorien suchen: Titel, Autor, Beschreibung, Verlag, Typ und ISBN-Nummer.

Warenkorb Der Warenkorb zeigt alle sich im Warenkorb befindlichen Dokumente an. Sie können von dort noch einige Einträge tätigen und von dort aus dann die Lizenzen kaufen.

Hilfe Die Hilfe wird im Layout 1 durch ein Fragezeichen dargestellt, sofern Sie auf dieses Fragezeichen klicken erscheint in einem neuen Browser Fenster die Hilfe. Wobei die Hilfe im Layout 1 abhängig vom Inhalt des Hauptframes ist, befindet sich im Hauptframe der Warenkorb und Sie klicken auf das Frasgezeichen, erscheint ein neues Fenster in dem Ihnen die Hilfe zum Warenkorb angezeigt wird.

Layout 2:

Abbildung 7.6: Das Layout 2

Im linken oberen Frame des Layouts 2 (Abb. 7.6), also dort wo sich das Logo befindet, gibt es unterhalb des Logos ein Pop-Up Menü „Auswahl“, das folgende Auswahlpunkte für die Navigation innerhalb des iShops zur Verfügung stellt:

Startseite Mit diesem Menüpunkt wird die ShopInfo-Seite aufgerufen. Dies ist entweder eine Seite, die vom Betreiber erstellt wurde und seinen Vorstellungen entspricht, oder aber es wird die Seite eingeladen, die von einem Servlet erstellt wird und die nötigsten Informationen über den Shop enthält.

allgemeine Hilfe Hier wird die allgemeine Hilfe zum WebShop aufgerufen und im Hauptframe dargestellt.

KundenDaten Hier gelangen Sie zu einem Fenster in dem Sie sich gegebenenfalls anmelden oder Ihre Nutzerdaten pflegen, sowie Ihr Kundenkonto aufladen und sich einen Kontoauszug ausgeben lassen können.

Suchen Ein Mausklick auf diesen Link führt Sie zur Suchseite, wo Sie die Möglichkeit haben, nach einem Suchbegriff im Datenbestand suchen zu lassen.

Lizenzen Es gibt zwei Arten von Lizenzen: einmal die gültigen Lizenzen und einmal die komplette Liste an Lizenzen; das sind noch die Lizenzen dazu, die noch nicht freigeschaltet sind. Unter diesem Menüpunkt werden erst die gültigen Lizenzen angezeigt, dann sind aber die kompletten Lizenzen mittels eines Buttons am unteren Ende der Seite anzeigbar..

Kontakt Mit diesem Menüpunkt wird die ShopInfo-Seite aufgerufen. Dies ist entweder eine Seite die vom Betreiber erstellt wurde und seinen Vorstellungen entspricht, oder aber es wird die Seite eingeladen, die von einem Servlet erstellt wird und die nötigsten Informationen über den Shop enthält.

Abbildung 7.7: Die Navigation durch die Dokumentkategorien im Layout 2

Im Layout 2 befindet sich im rechten oberen Frame, die Navigationsleiste durch die einzelnen Dokumentkategorien (Abb. 7.7), die vom aktuellen Online-Shop angeboten werden. Initial werden hier die Hauptkategorien geladen, sofern Sie dann auf eine Hauptkategorie klicken, steht im oberen Teil des Frames der Name der Hauptkategorie und im unteren Tei des Frames die Namen, der in dieser Kategorie enthaltenen Unterkategorien. Desweitern werden im Hauptframe die in dieser Kategorie enthaltenen Dokumente angezeigt.

Der Warenkorb befindet sich beim Layout 2 permanent im rechten unterern Frame. So haben Sie jederzeit einen Überblick über die aktuell im Warenkorb befindlichen Dokumente. Dies ist bei Layout 1 und 3 nicht der Fall, hier müssen Sie extra auf den Menüpunkt „Warenkorb“klicken, um sich dann im Hauptframe den Warenkorb anzeigen zu lassen.

Layout 3:

Abbildung 7.8: Das Layout 3

Im Layout 3 (Abb. 7.8) befindet sich wie bei den anderen Layouts auch im rechten oberen Eck das Logo des Online-Shops, wie bei Layout 1 existiert hier auch keine Interaktionsmöglichkeit.

Im Layout 3 befindet sich im linken unteren Frame, die Navigationsleiste (Abb. 7.9), die sich sowohl auf den Dokumentbestand als auch auf die Navigation durch den Shop bezieht. Diese Navigationsleiste beinhaltet folgende Menüpunkte.

Startseite Mit diesem Menüpunkt wird die ShopInfo-Seite aufgerufen. Dies ist entweder eine Seite, die vom Betreiber erstellt wurde und seinen Vorstellungen entspricht, oder aber es wird die Seite eingeladen, die von einem Servlet erstellt wird und die nötigsten Informationen über den Shop enthält.

Kategorien Hier können Sie sich durch die Dokumentkategorien, die vom aktuellen Online-Shop angeboten werden, hindurchnavigieren. Zu Beginn bekommen Sie nur die Hauptkategorien zu sehen. Diese können Sie dann aber durch anklicken weiter aufgeklappen. So ergibt sich dann eine komplette Baumstruktur durch die verschiedenen Kategorien. Klicken Sie auf ein Plus-Symbol wird der Baum weiter aufgeklappt, klicken Sie auf ein Minus-Symbol wird der Baum wieder eingeklappt und wenn Sie direkt auf den Namen

Abbildung 7.9: Die Navigation im Layout 3

einer Kategorie klicken, dann wird im Hauptframe die Dokumentanzeige eingeladen, diese listet alle Dokumente der Kategorie auf die Sie angeklickt haben.

Benutzer einloggen Hier gelangen Sie zur Login-Seite, wo Sie sich mit Ihrem Login und Passwort anmelden können und Ihnen stehen anschliessend weitere Unterpunkte in der Navigation zur Verfügung:

Kundendaten pflegen Hier können Sie sich Ihre Kundendaten anzeigen lassen und bei Bedarf Ihren Vorstellungen anpassen, bzw. bei nötigen Änderungen diese dort eintragen (z.B. Wohnortwechsel), desweiteren können Sie an dieser Stelle jederzeit Ihr Passwort ändern. Sie können allerdings zu keinem Zeitpunkt mehr Ihr Login ändern.

Lizenzen anzeigen Zeigt Ihre Momentan gültigen Lizenzen an, und sofern Sie auf den Link für „alle Lizenzen“klicken, auch alle Ihre Lizenzen.

Kundenkonto aufladen Unter diesem Link haben Sie jederzeit die Möglichkeit Ihr Kundenkonto aufzuladen.

Logout Hier können Sie sich als der aktuelle Benutzer ausloggen, falls Sie den Shop verlassen wollen oder aber nur Ihren Benutzer ändern wollen.

Warenkorb Der Warenkorb zeigt alle sich im Warenkorb befindlichen Dokumente an. Sie können dort noch einige Einträge tätigen und von dort aus dann Ihre Lizenzen kaufen.

Suchen Ein Mausklick auf diesen Link führt Sie zur Suchseite, wo Sie die Möglichkeit haben, nach einem Suchbegriff im Datenbestand suchen zu lassen.

Hilfe Die Hilfe ruft ein Fenster auf in dem die Hilfe zum aktuellen Fenster angezeigt wird, das heisst die Hilfe die zu dem Inhalt des rechten unteren Teils des Fensters gehört wird angezeigt.

ShopInfo Mit diesem Menüpunkt wird die ShopInfo-Seite aufgerufen. Dies ist entweder eine Seite, die vom Betreiber erstellt wurde und seinen Vorstellungen entspricht, oder aber es wird die Seite eingeladen, die von einem Servlet erstellt wird und die nötigsten Informationen über den Shop enthält.

7.3.2.3 Die verschiedenen Interaktionsfenster

Diese Fenster sind bis auf eine Ausnahme nicht abhängig vom Layout, und werden immer im Hauptframe dargestellt. Nur der Warenkorb wird im Layout2 nicht im Hauptframe angezeigt, sondern ist permanent im linken unteren Frame dargestellt. Der Aufbau des Warenkorbes unterscheidet sich nicht zu dem Aufbau im Layout 1 und 3, deswegen wird der Warenkorb allgemein an dieser Stelle erklärt.

Login:

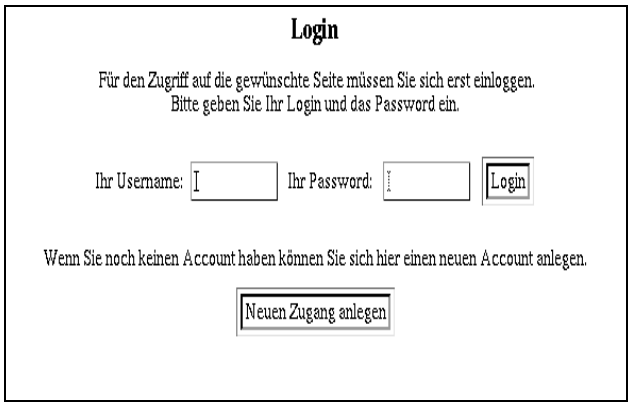
Login
Für den Zugriff auf die gewünschte Seite müssen Sie sich erst einloggen.
Bitte geben Sie Ihr Login und das Password ein.
Ihr Username:
Ihr Password:
Login
Wenn Sie noch keinen Account haben können Sie sich hier einen neuen Account anlegen.
Neuen Zugang anlegen

Abbildung 7.10: Das Login-Fenster

Für einige Aktionen ist es notwendig, dass der Benutzer sich vorher anmeldet. In einem solchen Fall wird ggf. das Login-Fenster (Abb. 7.10) vor dem eigentlichen Kontext dargestellt, dort müssen Sie sich dann mit Ihrem Login und Passwort anmelden. Falls Sie noch kein Login und Passwort besitzen, besteht dort noch für Sie die Möglichkeit sich einen eigenen Zugang anzulegen. Wenn Sie einen eigenen Zugang besitzen, geben Sie bitte zuerst das Login und dann das Passwort ein. Danach erfolgt die Weiterleitung zum entsprechenden Interaktionsfenster.

Warenkorb:

Der Warenkorb (Abb. 7.11) bietet Ihnen einen Überblick über Ihre eingepackten Dokumente und führt Sie auf die Bestellseite. Hier gelangen Sie zur Anzeige Ihres aktuellen Warenkorbs, in dem Sie noch weitere Einstellungen vornehmen können und für die enthaltenen Dokumente Lizenzen bestellen können. Der Warenkorb bietet die Möglichkeit zunächst mehrere Dokumente zwischen zu speichern, um dann später entscheiden zu können, ob man die entsprechenden Lizenzen kaufen möchte bzw. wie viele Lizenzen eines Dokuments man für welchen Zeitraum kaufen möchte.

Sowohl bei Layout 1 als auch bei Layout 3 befindet sich im Menü der Punkt Warenkorb. Wenn Sie auf Warenkorb klicken, werden Ihnen die Dokumente angezeigt die Sie in den Warenkorb gelegt haben. Sollten sich keine Dokumente im Warenkorb befinden, wird dies durch den folgenden Informationstext deutlich gemacht: „Ihr Warenkorb ist noch leer!“. Im Layout 1 wird dieser Text im Hauptframe angezeigt, im Layout 2 wird dieser Text schon zu Beginn im linken unteren Frame angezeigt, und im Layout 3 wird dieser Text oberhalb des Hauptframes angezeigt. Wenn sich Dokumente im Warenkorb befinden, werden sie auf folgende Weise aufgelistet:

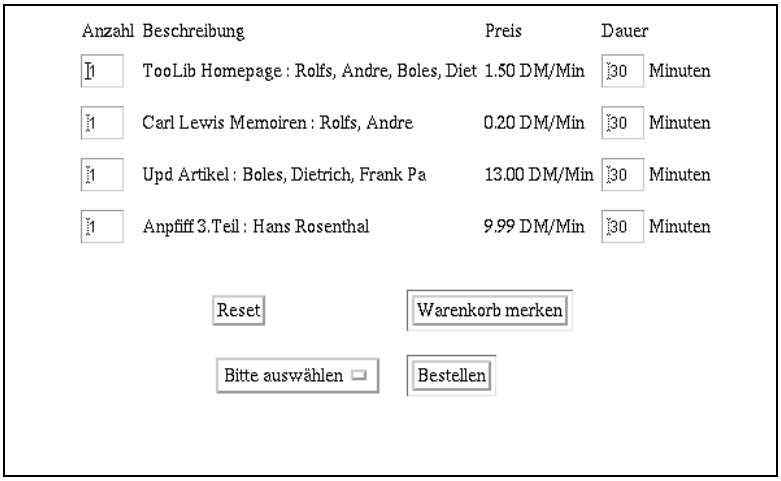
Anzahl Beschreibung Preis Dauer
1 TooLib Homepage : Rolfs, Andre, Boles, Diet 1.50 DM/Min 30 Minuten
1 Carl Lewis Memoiren : Rolfs, Andre 0.20 DM/Min 30 Minuten
1 Upd Artikel : Boles, Dietrich, Frank Pa 13.00 DM/Min 30 Minuten
1 Anpfiff 3.Teil : Hans Rosenthal 9.99 DM/Min 30 Minuten
Reset
Warenkorb merken
Bitte auswählen
Bestellen

Abbildung 7.11: Der Warenkorb

Anzahl: Die Anzahl der Lizenzen kann von Ihnen in diesem Eingabefeld abgeändert werden. Wenn Sie das Dokument nicht mehr bestellen möchten, setzen Sie einfach die Anzahl auf Null, allerdings verschwindet das Dokument dadurch nicht aus dem Warenkorb.

Kurzbeschreibung: Eine Kurzbeschreibung des Dokuments bestehend aus Titel und Autor(en).

Preis/Minute: Der Preis für eine Lizenz in DM/Minute. Dieses Feld kann natürlich von Ihnen nicht verändert werden.

Dauer [Minuten :] Für wieviele Minuten Sie Lizenzen dieses Dokuments kaufen möchten, können Sie in dieses Eingabefeld eintragen.

Zusätzlich stehen Ihnen noch die folgenden Funktionalitäten im unteren Bereich des Warenkorbs in Form von Buttons zur Verfügung:

Reset: Durch einen Mausklick auf den Reset-Button werden alle Werte in den Eingabefeldern wieder auf den letzten gesicherten Wert zurückgesetzt.

Warenkorb merken: Ein Mausklick auf diesen Button bewirkt, dass die gerade aktuellen Werte aus den Eingabefeldern gespeichert werden und somit bei Ihrem nächsten Mausklick auf den Reset-Button diese Werte wieder angenommen werden. Auch nach dem Ausloggen bleiben die gespeicherten Werte des Warenkorbs bestehen.

Auswahl einer Zahlungsmethode: Hier können Sie bereits an dieser Stelle schon eine Zahlungsmethode für Ihre Bestellung auswählen, wobei Sie die Wahl zwischen den konventionellen Zahlungsmethoden wie Bankeinzug oder Creditcard und den moderneren Zahlungsmethoden wie CyberCash, ECash oder der Zahlung per Kundenkonto haben. Sie klicken einfach auf die „Bitte auswählen“-Auswahlbox, wobei ein Pop-Up-Menü aufklappt, in dem Sie eine dieser Zahlungsmethoden auswählen können.
Hierbei ist noch anzumerken, dass es möglich ist, dass nicht alle genannten Zahlungsmethoden vom WebShop angeboten werden!

Bestellen: Ein Mausklick auf den Bestellen-Button führt Sie auf die Bestellseite, wo ggf. noch weitere Eingaben verlangt werden. Falls Sie noch nicht angemeldet sind, müssen Sie sich zuvor erst anmelden; dazu erscheint das weiter oben erwähnte Login-Fenster im Hauptframe.

Benutzerdaten anlegen/ ändern:

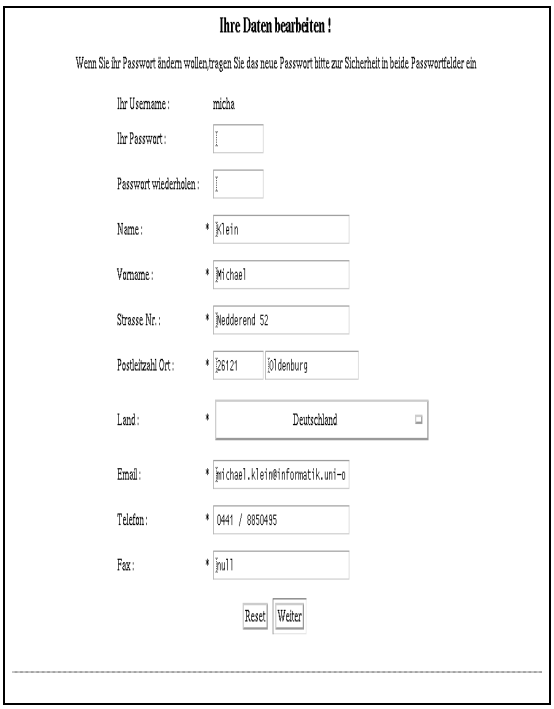

Abbildung 7.12: Das Benutzerdaten-Fenster

Hier (Abb. 7.12) können Sie sich gegebenenfalls anmelden oder Ihre Nutzerdaten pflegen, sowie Ihr Kundenkonto aufladen und sich einen Kontoauszug ausgeben lassen. Sie können allerdings nicht Ihr Login ändern; dieses kann von Ihnen nur einmal (während der Anmeldung für den iShop) gewählt werden, Ihr Passwort können Sie aber jederzeit ändern.

Hilfe:

Die kontextsensitive Hilfe (Abb. 7.13) wird, je nach Inhalt des Hauptframes, in einem extra Fenster angezeigt. Wenn zum Beispiel im Hauptframe Ihre Lizenzen angezeigt werden, so erscheint im Hilfefenster die Hilfeseite zur Lizenzanzeige. Sie erreichen die kontextsensitive Hilfe im Layout 1, in dem Sie einfach auf das Fragezeichen in der Navigation klicken. Im Layout 2 und Layout 3 gelangen Sie mit einem Klick auf das kleine Fragezeichen-Symbol, welches Ihnen des Öfteren in den Anzeigen begegnen wird, zur kontextsensitiven Hilfe. Innerhalb des neuen Hilfefensters können Sie mittels der Navigationshilfen im oberen und unteren Bereich navigieren. Unter dem Link Index verbirgt sich ein kurzes Inhaltsverzeichnis der Hilfe hier können Sie also schnell ein spezielles Thema auswählen über das Sie informatiert werden wollen..

Suchen:

Das Suchen-Frame (Abb. 7.14) bietet Ihnen die Möglichkeit einer Suche auf den Dokumenten des iShops. Ein Mausklick auf diesen Link führt Sie zur Suchseite, wo Sie die Möglichkeit haben, nach einem Suchbegriff im Datenbestand suchen zu lassen. Die Seite für die Suche bietet Ihnen die Möglichkeit auf den Dokumenten in der Datenbank eine Suche nach den folgenden derzeitigen Kriterien durchzuführen:

Titel Der Titel des Dokumentes nach dem Sie suchen.

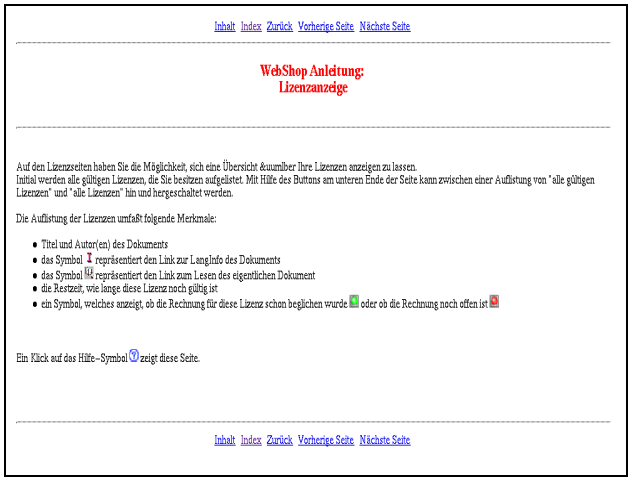
Inhalt Index Zurück Vorherige Seite Nächste Seite
WebShop Anleitung:
Lizenzanzeige
Auf den Lizenzseiten haben Sie die Möglichkeit, sich eine Übersicht &uumlber Ihre Lizenzen anzeigen zu lassen.
Initial werden alle gültigen Lizenzen, die Sie besitzen aufgelistet. Mit Hilfe des Buttons am unteren Ende der Seite kann zwischen einer Auflistung von "alle gültigen Lizenzen" und "alle Lizenzen" hin und hergeschaltet werden.
Die Auflistung der Lizenzen umfaßt folgende Merkmale:
Titel und Autor(en) des Dokuments
das Symbol repräsentiert den Link zur LangInfo des Dokuments
das Symbol repräsentiert den Link zum Lesen des eigentlichen Dokument
die Restzeit, wie lange diese Lizenz noch gültig ist
ein Symbol, welches anzeigt, ob die Rechnung für diese Lizenz schon beglichen wurde oder ob die Rechnung noch offen ist
Ein Klick auf das Hilfe–Symbol zeigt diese Seite.
Inhalt Index Zurück Vorherige Seite Nächste Seite

Abbildung 7.13: Das Hilfe-Fenster

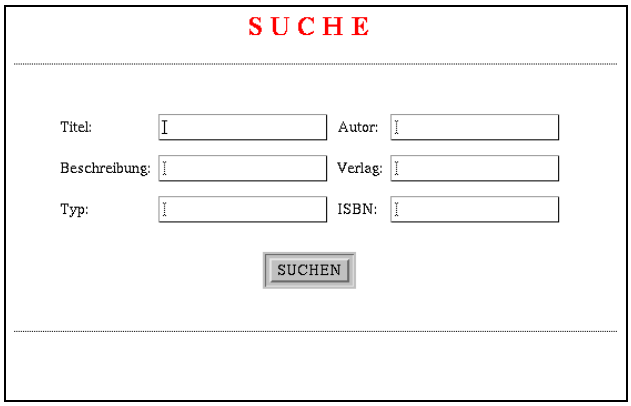

Abbildung 7.14: Das Suchen-Fenster

Autor Einer der Autoren der bei der Bearbeitung des Dokumentes mitgeholfen hat.

Beschreibung Die Beschreibung, hier können Sie Schlagwörter eingeben die in den gefundenen Dokumenten vorkommen sollen.

Verlag Falls Sie nach den Dokumenten eines bestimmten Verlages suchen.

Typ Welche Art von Dokument suchen Sie.

ISBN-Nummer Die ISBN-Nummer des Dokumentes.

Im Suchformular haben Sie die Möglichkeit in die Suchfelder Ihre Suchbegriffe einzutragen, nach denen dann in der Datenbank gesucht werden soll. Die angegebenen Suchbegriffe werden mittels einer UND-Verknüpfung als Suchanfrage an die Datenbank aufgefasst. Ihre Suchanfrage schicken Sie mit einen Mausklick auf den SUCHEN-Knopf ab. Als Ergebnis bekommen Sie eine Liste mit Dokumenten, die auf Ihre gestellte Suchanfrage passen. Werden in die Suchfelder keine Suchbegriffe eingetragen, werden alle vorhandenen Einträge aus der Datenbank als Ergebnis

zurückgeliefert. Auf den Ergebnissen bestehen die gleichen funktionellen Möglichkeiten wie bei der Kurzinfoanzeige auf der Seite mit der Auflistung der Dokumente einer Kategorie.

Die ShopInfo-Seite:

Abbildung 7.15: Das ShopInfo - Fenster

Hier (Abb. 7.15) werden Ihnen in kompakter Form die wichtigsten Informationen über den Shop und den Shopbetreiber angezeigt. Dies reicht von den Kontaktmöglichkeiten bis hin zu Informationen zu den akzeptierten Zahlungsmethoden im Shop und den allgemeinen Geschäftsbedingungen. Diese Seite wird ebenfalls von den Servlets erstellt sofern der Betreiber des Shop hier keine eigenen Seiten eingibt.

Die Start-Seite Die Start-Seite kann vom Betreiber im Authoring-Tool angegeben werden. Sollte der Betreiber allerdings keine Angaben zu einer Sart-Seite gemacht haben, so können die Servlets selbst eine Seite erstellen, die dann allerdings der ShopInfo-Seite (siehe 7.15) entspricht.

Die Kontakt-Seite (nur für Layout 2) Die Kontaktseite des Shop-Betreibers entspricht der ShopInfo-Seite aus Layout 1 und 3 (siehe Abschnitt „Die ShopInfo-Seite“ oder „Die Start-Seite“). Der Menüpunkt “Kontakt„ist nur im Layout 2 im Pop-Up-Menü “Auswahl„zu finden.

Die Lizenz-Anzeige:

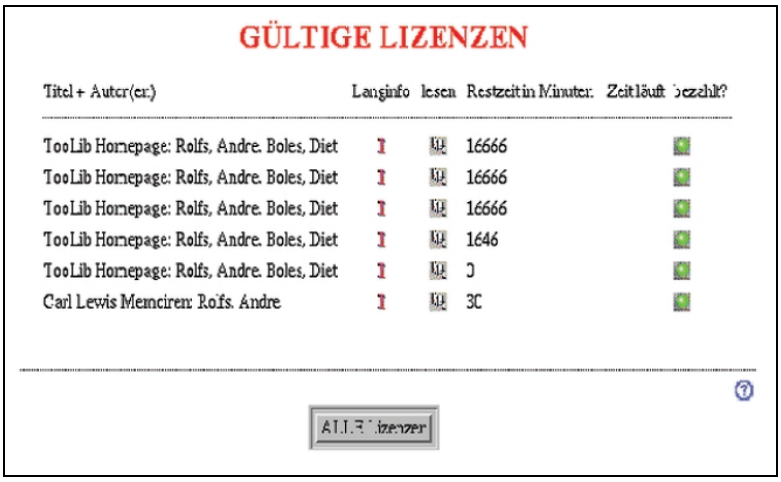

Abbildung 7.16: Das Lizenz-Anzeige

Es gibt zwei verschiedenen Lizenz-Anzeigemöglichkeiten (Abb. 7.16). Sofern Sie auf den Menüpunkt Lizenzen klicken, erscheinen alle Ihre Lizenzen, die momentan gültig sind. Mit Hilfe des Buttons am unteren Ende des Haupframes gelangen Sie dann zur Auflistung aller Ihrer Lizenzen und von dort mit dem selben Button wieder zurück zur Anzeige aller gültigen Lizenzen. Die Lizenzanzeige ist selbstverständlich erst möglich, wenn Sie sich ale Benutzer des Online-Shops eingeloggt haben (hierzu siehe weiter oben).

Die Auflistung der Lizenzen umfasst folgende Merkmale:

Titel Der Titel des Dokuments.

Autor(en) Die Autoren die an der Entstehung des Dokuments mitgewirkt haben.

Das Infosymbol Dises Symbol repräsentiert einen Link auf die Langinfo Seite eines Dokuments.

Das Lesesymbol Das zweite Symbol repräsentiert den Link zum Lesen des eigentlichen Dokuments.

Restzeit Zeit, wie lange diese Lizenz noch gültig ist.

Das letzte Symbol Der rote oder grüne Punkt gibt an, ob eine Lizenz freigeschaltet wurde, oder ob diese noch nicht freigeschaltet ist.

Langinfo eines Dokuments:

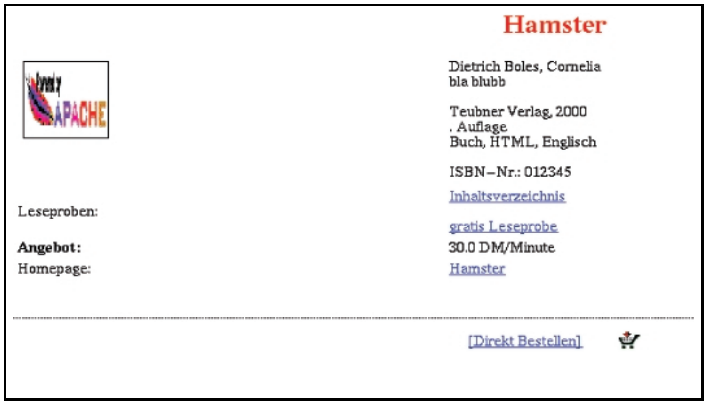

Abbildung 7.17: Die Langinfo zu einem Dokument

In der Dokumentenansicht wird ein Dokument näher beschrieben. Die Langinfo (Abb. 7.17) bietet Ihnen die Möglichkeit, sich ausgiebig über das ausgewählte Dokument zu informieren. Hierzu wird - falls vorhanden - das Cover des Dokuments angezeigt, Sie können einen Blick in das Inhaltsverzeichnis werfen, ggf. eine gratis Leseprobe runterladen oder anschauen, oder sich auf der Homepage, die zu dem Dokument existiert, noch näher informieren. Desweiteren haben Sie die Möglichkeit das Dokument in ihren Warenkorb zu legen oder sogar dirket zu bestellen. Hierfür sind die Buttons „Warenkorb“(durch realisiert) und „Direkt Bestellen“gedacht.

Konto-Anzeige:

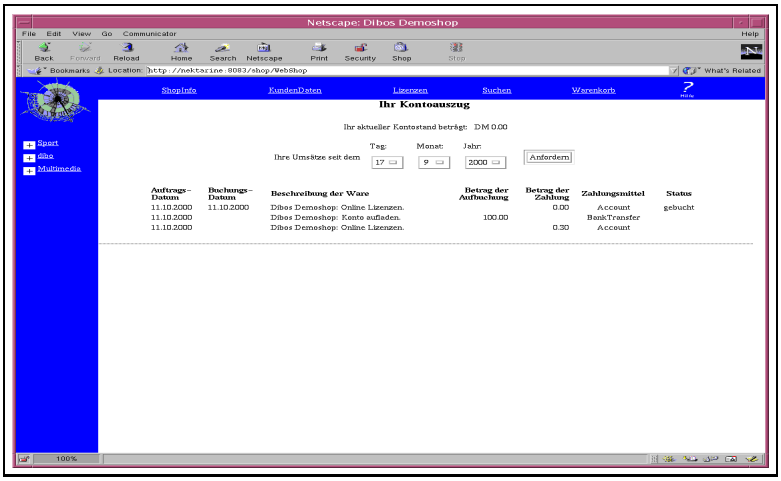

Abbildung 7.18: Das Kundenkonto-Fenster

Auf der Seite der Kontoanzeige (Abb. 7.18) werden Ihnen Ihre Transaktionen im WebShop in Form einer Tabelle aufgelistet. Nach der Anzeige Ihres Namens und Ihrer Anschrift, die aus den Einträgen Ihrer Kundendaten übernommen wurden, wird Ihnen angezeigt, welcher Betrag Ihnen noch auf Ihren Kundenkonto zur Verfügung steht.

Im mittleren Abschnitt steht Ihnen ein Formular zum Ändern des Zeitraums Ihres Kontoauszugs zur Verfügung, falls der vom Programm gewählte Bereich für Ihre Belange zu eingeschränkt ist.

Wenn Sie den Zeitraum Ihres Kontoauszugs ändern möchten, verwenden Sie bitte das Formular, um das Datum einzustellen, ab dem Sie Ihren Kontoauszug angezeigt bekommen möchten. Anschliessend klicken Sie auf den SUBMIT-Button.

Im unteren Teil des Kontoauszugs werden Ihnen in einer Tabelle mit den folgenden Spalten Ihre getätigten Transaktionen im WebShop aufgelistet. Dabei beginnt die Auflistung erst ab dem im Formular angegebenen Datum.

Buchung: Das Datum Ihrer Buchungen, beginnend beim letzten Termin.

Bezahlung: Das Datum Ihrer Bezahlung des gewünschten Dokuments.

Beschreibung der Ware: Eine kurze Beschreibung über die von Ihnen bestellte Ware.

Betrag der Aufbuchung: Der Betrag, der auf Ihr Kundenkonto aufgebucht wurde.

Betrag der Zahlung: Der Betrag, der von Ihnen bezahlt wurde.

Zahlungsmittel: Das von Ihnen benutzte Zahlungsmittel (Kundenkonto, Kreditkarte, E-Cash,...)

Status: Der Status, ob die Bezahlung bzw. Aufbuchung schon durchgeführt wurde bzw. schon abgeschlossen ist.

Dokumentanzeige:

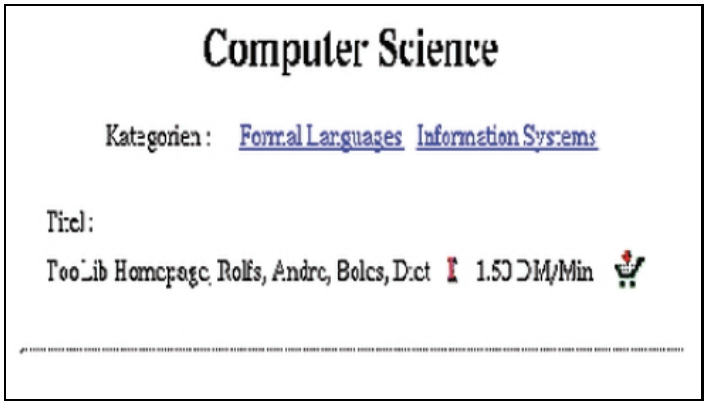
Computer Science
Kategorien: Formal Languages Information Systems
Titel:
Too_lib Homepage, Rolfs, Andre, Bolcs, Dict 1.50 DM/Min

Abbildung 7.19: Die Dokumentanzeige

Im Hauptframe werden - falls vorhanden - die Dokumente aus der gewählten Kategorie in Form einer Kurzfassung dargestellt (Abb. 7.19). Dabei werden für jedes Dokument die folgenden Merkmale angezeigt bzw. in Form von kleinen Grafiken dargestellt:

Titel des Dokuments Unter welchem Titel das Dokument erschienen ist.

Autor(en) des Dokuments Wer an der Entwicklung des Dokumentes beteiligt war.

Das Infosymbol I Diese Bild repräsentiert einen Link auf die LangInfo zu dem entsprechenden Dokument.

Preis Der Preis für eine Lizenz über diese Dokument in DM pro Minute.

Das Warenkorbsymbol Diese Bild erscheint nur, wenn das Dokument noch nicht im Warenkorb enthalten ist.

Sie haben nun die Möglichkeit sich durch einen Mausklick auf das LangInfo-Symbol dieselbige anzuschauen. Desweiteren steht ihnen ggf. noch der Link zur Verfügung, um dieses Dokument in den Warenkorb legen zu können.

7.3.3 Die Interaktionsmöglichkeiten des Benutzers

Hier werden einzelne Aktionen, die Sie ebenfalls anwenden müssen, wenn Sie den Shop benutzen wollen, erklärt. Hier können Sie bei der späteren Benutzung des Programms Hilfestellung finden, wenn sich irgendwelche Probleme finden.

7.3.3.1 Auswählen eines Dokuments

Um sich ein Dokument auszuwählen müssen Sie sich ersteinmal für eine Kategorie entscheiden aus der Ihr gesuchtes Dokument stammen soll. Oder aber Sie benutzen die Suche, um sich ein bestimmtes Dokument anzeigen zu lassen. Wenn Sie dann über die Kurzinfo zu dem gewünschten Dokument gelangt sind, haben sie die Möglichkeit sich die Langinfo anzeigen zu lassen, es in den Warenkorb zu legen oder aber es uas der Langinfo heraus direkt zu bestellen.

7.3.3.2 Einen neuen Benutzer anlegen

Immer wenn personenbezogene Daten aus der Datenbank benötigt werden und Sie noch nicht eingeloggt sind, wird die Login-Seite aufgerufen, damit Sie sich als rechtmässiger Nutzer identifizieren können.

Dann werden Sie zur Eingabe Ihres Usernamens und Ihres persönlichen Passworts aufgefordert. Geben Sie Ihren Usernamen und Ihr Passwort in die beiden Eingabefelder ein und klicken Sie mit der Maus auf den Login-Button, um sich einzuloggen. Falls die Eingaben falsch waren, erscheint die Login-Seite erneut mit einer kurzen Fehlermeldung und Sie können es noch einmal versuchen.

Falls Sie noch keinen eigenen Usernamen und Passwort besitzen, können Sie sich durch einen Klick mit der Maus auf den „Neuen Zugang anlegen “-Button einen neuen Account anlegen.

7.3.3.3 Die persönlichen Daten ändern

Durch anklicken des Menüpunktes „Benutzerdaten ändern/ anlegen “gelangen Sie in das Frame, wo Sie die Möglichkeit haben Ihre Kundendaten zu ändern. Folgende Angaben können sie ändern:

- Passwort
- E-Mail Adresse
- Adresse (dazu gehören Länderwahl, Wohnort, Straße, Hausnummer)
- Telefon
- Fax

Nicht ändern können Sie Ihr Login.

7.3.3.4 Lizenzen zu Dokumenten kaufen

Sie haben zwei Möglichkeiten, aus dem WebShop heraus, Lizenzen zu bestellen:

Aus dem Warenkorb heraus: Durch Klicken auf den „Bestellen-Knopf"im Warenkorb gelangen Sie direkt auf die Bestellseite. Von dort aus können Sie dann Ihre Dokumente bestellen.

Direkt Bestellen: Durch einen Mausklick auf den Link „Direkt Bestellen", auf der Seite mit der Langinfo zu einem Dokument, gelangen Sie auf eine Seite, die dem Warenkorb sehr ähnlich sieht und auf der Sie die Anzahl und die Dauer der Lizenz zu dem Dokument einstellen können. Ein Mausklick in diesem Formular auf „Bestellen"führt Sie auf die Bestellseite, die im Rahmen der Bestellung weiter unten näher erläutert wird.

7.3.3.5 Suche nach einem Dokument

Sie klicken auf den Menüpunkt „Suchen"und gelangen automatisch in das Suchenfenster. Dort geben Sie Ihren Suchwunsch ein und erhalten, sofern vorhanden, ein Ergebnis das genauso aussieht wie die Kurzinfo zu den Dokumenten.

7.3.3.6 Benutzung der Hilfe

Der Menüpunkt „Hilfe"zeigt Ihnen die Hilfe an, hierbei gibt es einige Unterschiede zu den verschiedene Layouts. Bei Layout 1 wird die Hilfe grundsätzlich in einem extra Fenster angezeigt, das vom Browser geöffnet wird. Bei den beiden anderen Layouts wird die allgemeine Hilfe im Hauptframe angezeigt. Die kontextsensitive Hilfe ist beim Layout 2 und Layout 3 durch anklicken auf das kleine Fragezeichen-Symbol zu erreichen und wird dann in einem extra Fenster angezeigt. Wenn z.B. der Warenkorb angezeigt wird und Sie klicken auf Hilfe bzw. auf das Fragezeichen-Symbol, so erscheint die Hilfe zum Warenkorb im Hilfefenster. Innerhalb des Hilfeframes können Sie sich durch die verschiedenen Punkte manövrieren oder das Inhaltsverzeichnis über den Link Index erreichen.

7.3.3.7 Bestellung

Auf der Bestellseite wird Ihnen Ihre Bestellung noch einmal aufgeführt und zusammen mit dem Gesamtpreis angezeigt. Falls Sie noch etwas an den Einträgen ändern möchten, haben Sie die Möglichkeit noch einmal zurück zu gehen bzw. im Warenkorb die notwendigen Änderungen vorzunehmen. Im unteren Teil der Bestellseite haben Sie die Möglichkeit die Zahlungsmethode auszuwählen und das Bestellformular entsprechend der ausgewählten Zahlungsmethode anzupassen. Als Zahlungsmethoden stehen Ihnen hier die vom Shopbetreiber unterstützten Zahlungsmethoden aus denen, im Folgenden aufgeführten Methoden zur Verfügung:

Nachdem Sie Ihre Daten in das entsprechende Bestellformular eingegeben haben und Ihre Bestellung abgeschickt haben, erhalten Sie eine Nachricht über den Status Ihrer Bestellung - also ob die Transaktion geklappt hat, oder nicht.

Bestellung per Bankeinzug Im Bestellformular für die Zahlung per Bankeinzug müssen Sie die folgenden selbstbeschreibenden Felder ausfüllen:

- Kontonummer
- Bankleitzahl
- Kreditinstitut
- Name des Kontoinhabers: initial der Name des angemeldeten Nutzers

Im oberen Bereich werden Ihre Bestelldaten angezeigt und Sie haben die Möglichkeit, die Zahlungsmethode noch mal zu ändern oder anzupassen. Desweiteren wird Ihnen unterhalb des Formulars noch mal in einem kurzen Text mitgeteilt, wieviel DM von dem angegebenen Konto per Lastschrift eingezogen werden. Ein Mausklick auf „Bestellen“führt Ihre Bestellung aus, sofern keine weiteren Probleme aufgetreten sind.

Bestellung per Credit Card Im Bestellformular für die Zahlung per Credit Card müssen Sie die folgenden sebstbeschreibenden Felder ausfüllen:

- Kreditkartennummer
- Gültigkeitsmonat und -jahr Ihrer Kreditkarte
- Kreditkarteninstitut (z.B. Eurocard, VISA...)
- Name des Kreditkarteninhabers: initial der Name des angemeldeten Nutzers

Im oberen Bereich werden Ihre Bestelldaten angezeigt und Sie haben die Möglichkeit, die Zahlungsmethode noch mal zu ändern oder anzupassen. Desweiteren wird Ihnen unterhalb des Formulars noch mal in einem kurzen Text mitgeteilt, wieviel DM von Ihrem Kreditkartenkonto abgebucht werden. Ein Mausklick auf „Bestellen“führt Ihre Bestellung aus, sofern keine weiteren Probleme aufgetreten sind.

Bestellung per Kundenkonto Im oberen Bereich werden Ihre Bestelldaten angezeigt und Sie haben die Möglichkeit, die Zahlungsmethode noch mal zu ändern oder anzupassen. Für die Zahlung per Kundenkonto müssen Sie keine weiteren Daten eingeben, sondern lediglich die folgenden Bedingungen erfüllen:

- 1. Sie müssen ein Kundenkonto besitzen und
- 2. auf dem Kundenkonto muss genügend Geld vorhanden sein, um den Rechnungsbetrag bezahlen zu können. Ggf. müssen Sie vor der Bestellung das eigene Kundenkonto aufladen.

Desweiteren wird Ihnen unterhalb des Formulars noch mal in einem kurzen Text mitgeteilt, wieviel DM von Ihrem Kundenkonto abgebucht werden. Ein Mausklick auf „Bestellen“führt Ihre Bestellung aus, sofern keine weiteren Probleme aufgetreten sind.

Bestellung per ECash Im oberen Bereich werden Ihre Bestelldaten angezeigt und Sie haben die Möglichkeit, die Zahlungsmethode noch mal zu ändern oder anzupassen. Für die Zahlung per ECash müssen Sie keine weiteren Daten eingeben, sondern Sie benötigen eine eigene ECash-Wallet auf Ihrem Rechner, um die Transaktion ausführen zu können. Für nähere Informationen siehe den Teil des Berichts der Payment-Gruppe. Desweiteren wird Ihnen unterhalb des Formulars noch mal in einem kurzen Text mitgeteilt, wieviel DM per ECash aus der Benutzerwallet eingezogen wird. Ein Mausklick auf „Bestellen“führt Ihre Bestellung aus, sofern keine weiteren Probleme aufgetreten sind.

Bestellung per CyberCash Im oberen Bereich werden Ihre Bestelldaten angezeigt und Sie haben die Möglichkeit, die Zahlungsmethode noch mal zu ändern oder anzupassen. Für die Zahlung per CyberCash müssen Sie keine weiteren Daten eingeben. Sie benötigen lediglich die CyberCash-Software. Für nähere Informationen über das Bezahlen per CyberCash klicken Sie bitte mit der Maus auf „Infos zu CyberCash“weitere Infos zu CyberCash (siehe auch Doku der DDS-Payment-Gruppe). Desweiteren wird Ihnen unterhalb des Formulars noch mal in einem kurzen Text mitgeteilt, wieviel DM per CyberCash eingezogen werden. Ein Mausklick auf „Bestellen“führt Ihre Bestellung aus, sofern keine weiteren Probleme aufgetreten sind.

7.3.3.8 Kundenkonto aufladen

Über den Menüpunkt Kundendaten in der Navigationsleiste und einen Mausklick auf den Link Kundenkonto aufladen gelangen Sie zur Seite, um Ihr Kundenkonto wieder aufzuladen. Sie tragen in das Formular einen Betrag ein, womit Ihr Kundenkonto aufgeladen werden soll, wählen eine Zahlungsmethode aus der Auswahlliste aus und führen die Transaktion mit einem Mausklick auf Bestellen durch. Es versteht sich natürlich von selbst, dass Ihr Kundenkonto nicht mit der Zahlungsmethode Kundenkonto aufgeladen werden kann.

7.4 Klassenstruckturen

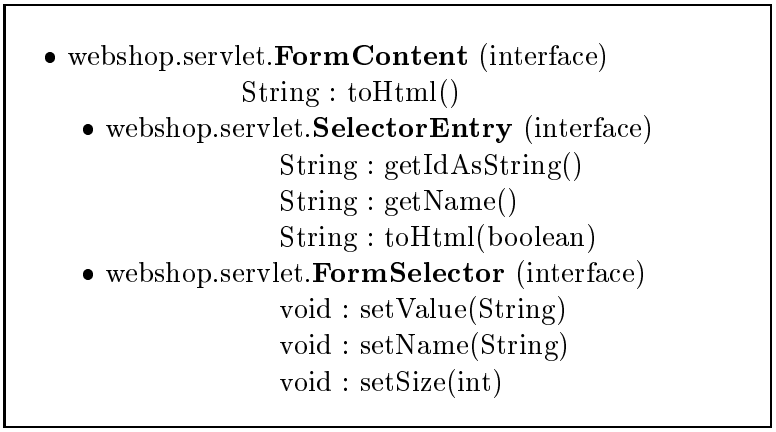

Abbildung 7.20: Die Interfaces

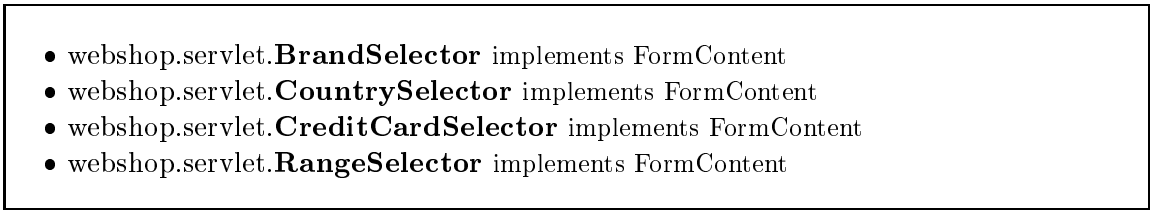

Abbildung 7.21: Die Klasssen die FormSelector implementieren

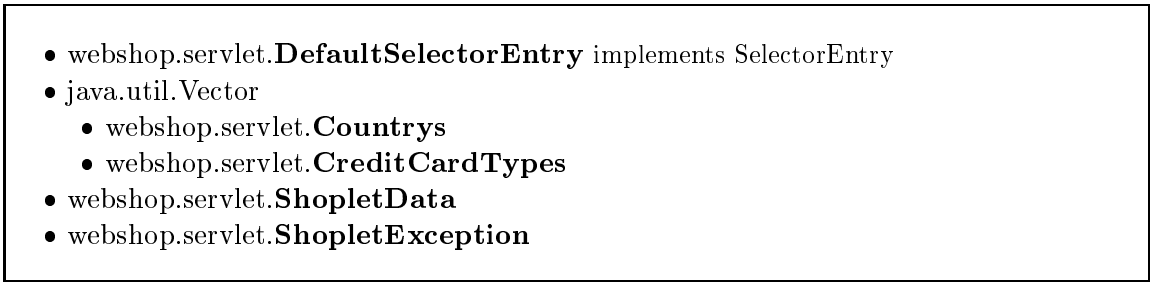

Abbildung 7.22: sonstige Klasssen

- webshop.classes.**BankAccount**
  - webshop.servlet.**BankAccountFormContent** implements FormContent
    BankAccountFormContent(HttpServletRequest)
    String : toHtml()
- webshop.classes.**CreditCard**
  - webshop.servlet.**CreditCardFormContent** implements FormContent
    CreditCardFormContent(HttpServletRequest)
    String : toHtml()
- webshop.servlet.**CustomerForm** implements FormContent
  CustomerForm(HttpServletRequest)
  CustomerForm(webshop.classes.Customer)
  CustomerForm()
  String : toHtml()
  String : toHTML()
  String : toHTML(boolean)
  String : checkUpdate()
  String : checkCreate()
  webshop.classes.Customer : getCustomer()
  void : updateCustomer()
  String : get*()
  void : set*(String)
- java.util.Date
  - webshop.servlet.**DateFormContent** implements FormContent
    DateFormContent(HttpServletRequest)
    DateFormContent(java.util.Date)
    DateFormContent()
    String : toHtml()
- webshop.servlet.**FormInput** implements FormContent
  FormInput(String, String)
  String : toHtml()
  void : set*(String)
  - webshop.servlet.**HiddenInput**
    FormInput(String, String)
- webshop.classes.**DocumentInfo**
  - webshop.servlet.**DocumentInfoFormContent** implements FormContent
    String : toHtml()
    String : toHtml(boolean)
- webshop.classes.**SearchCriteria**
  - webshop.servlet.**SearchCriteriaFormContent** implements FormContent
    SearchCriteriaFormContent(HttpServletRequest)
    String : toHtml()
- webshop.payment.**Transaction**
  - webshop.servlet.**TransactionFormContent** implements FormContent
    String : toHtml()
- webshop.servlet.**LoginForm** implements FormContent
  FormInput(HttpServletRequest)
  String : toHtml()
  String : getLogin()
  String : getPassword()
  void : setTarget(String)

Abbildung 7.23: Die Klasssen die FormContent implementieren

Kapitel 8

Bewertung und Ausblick

Mit dem DDS-System haben wir ein System entwickelt, das es in seiner derzeitigen Ausbaustufe ermöglicht, iShops aufzubauen, über die ohne großen Aufwand hypermediale Dokumente kostenpflichtig angeboten werden können. Kunden können zum Online-Zugriff auf die Dokumente Lizenzen erwerben, die eine bestimmte Zeitspanne lang gültig sind. Weiterhin haben wir speziell auf den Information Commerce ausgerichtete Geschäftsmodelle und Sicherheitskonzepte analysiert und werden sie in das DDS-System integrieren. Ziel unserer Aktivitäten ist es, ein flexibel konfigurierbares und erweiterbares 'Testbett' insbesondere für Autoren und Verlage zu schaffen, über das diese Erfahrungen im Bereich des Information Commerce sammeln können. Weiterhin versuchen wir, mit Anbietern kommerzieller eShop-Systeme in Kontakt zu kommen, um die von uns entwickelten Konzepte in deren inzwischen weit verbreitete Systeme zu integrieren. An dieser Stelle sollen noch einige Punkte explizit genannt werden, die bei dieser Realisierung des DDS-Systems nicht ausreichend genug untersucht bzw. nicht integriert wurden und bei einer zukünftigen Weiterentwicklung des DDS-Systems beachtenswert wären. Auf der Seite des DDS-Authoring-Tools wäre es denkbar und sicherlich auch sinnvoll, einen Mehrbenutzerbetrieb zu realisieren. Gerade bei großen Datenbeständen erleichtert eine Aufteilung der Zuständigkeitsbereiche auf verschiedene Personen die Arbeit des Einzelnen und erhöht die Effizienz des Ganzen. Auch ist in dieser Version des DDS-Systems nur eine einzige Form des Erwerbes von digitalen Waren möglich, nämlich die direkte Online-Nutzung mittels eines Browsers, für eine Person zu einem festen Minutenpreis. Alternativen wie der Download von digitalen Waren, das Einrichten von Gruppenkonten und die Vergabe von Rabatten wurden zwar analysiert, jedoch nicht integriert. Auf zusätzliche Features wie das Erstellen von Statistiken oder Kundenprofilen wurde gänzlich aus Zeitgründen verzichtet. Darüber hinaus existieren noch viele weitere Punkte, deren Realisierung für einen komfortablen Umgang mit dem DDS-Sytsem wünschenswert wäre, aber wir hoffen, dass mit diesem System eine gute Basis im Bereich iShop-Systeme geschaffen wurde.

Index

AccountPaymentClient, 59
Anfragebehandlung, 78
Angebote-Fenster, 93
    Handbuch, 112
Artikel-Inhalte-Fenster, 94
    Handbuch, 110
Ausnahmen, *siehe* Fehlerklassen
Authoring Tool, 8
Autoren-Fenster, 93
    Handbuch, 111

Banküberweisung, 49
BankTransferPaymentClient, 59
Benutzerdaten
    ändern, 145
    anlegen, 145
    anlegen/ ändern, 137
Benutzerhandbuch
    Web-Komponente, 124
Bestellung, 146
    Bankeinzug, 147
    Credit Card, 147
    CyberCash, 148
    ECash, 148
    Kundenkonto, 147
Bezahl-Server, 49
bookPayment(), 53, 57
Brand, 53, 58

C-Modul, 78
CreditCardSSLPaymentClient, 59
CustomerDoesNotExistException, 59
CyberCash, 49, 61
    CashRegister, 61
CyberCashClient, 60

Dateimanager, 94, 102
    Handbuch, 112
Datenbank
    JDBC, 8
Datenbank-Fenster
    Autoren, 95
    Formate, 95
    Handbuch, 108
    Sprachen, 96
    Typen, 96
    Verlage, 95
DB-Schnittstellen, *siehe* webshop.dbif, 45
    DDS, 28
    Optimierungsaspekte, 47
    Payment-Handler, 40
DDS (Digital-Document-Shop), 7
DDS-Web-Komponente, 117
Detail-Fenster, 92
    Handbuch, 109
Digitale Bibliotheken, 8
Dokument
    ändern, 91, 114
    anlegen, 90, 114
    löschen, 90, 115
    verschieben, 115
Dokumentanzeige
    Kurz, 143
Dokumentauswahl, 145
Dokumente
    digitale, 9
    hypermediale, 9
Dokumentinfo
    Lang, 142

ECash, 49, 61
ECashPaymentClient, 60
elektronische Zahlungsverfahren, 49, 61
elektronische Zahlungsvorgänge, 10
Entitäten, 15
    DDS, 17
    Payment-Handler, 18
ER-Schemata
    DDS, 15
    Payment-Handler, 18

Fehlerklassen, *siehe* webshop.dbif Fehlerbehandlung
FormContent, 120

getBalance(), 51, 57
getBrandIds(), 50, 56
getCredit(), 52, 57
getCustomerTransactionsSinceDate(), 50
getErrorStatus(), 51, 57
getTaStatus(), 51, 57
getTransaction(), 50, 57

Haupt-Fenster, 102
    Handbuch, 105
Hauptfenster, 92
HBCI, 50
Hilfe, 137, 146
Holderkonzept
    Allgemeines, 85
    Alternativen, 88
    GUI-Klassenmodell, 86
    Vorteile, 87

IllegalParameterException, 59
Implementierung
    Authoring-Tool, 99
Informationen, 7
    digitale, 7
Informationsschutz, 7
insertCustomer(), 57
insertPaymentCustomer(), 53
Installation der Payment-Komponente, 61
invoiceRequest(), 52, 57
InvoiceResult, 54, 58
iShop, 8

Java
    Servlet, 11
JavaScript, 12
JDBC, 10

Klassen, *siehe* webshop.dbif
Klassenhierarchie des Payment-Systems, 56
Klassenstrukturen
    Web-Komponente, 149
Komponente
    Authoring, 10
    Datenbank, 10
    Dokumentenschutz, 10
    Payment, 10
    Web, 11
Konfigurations-Datei
    Authoring-Tool, 104
Kontoanzeige, 142
Kreditkarte, 49
Kundenkonto, 49
Kundenkonto aufladen, 148

Layout 1, 126
Layout 2, 129
Layout 3, 131
Layout-Ansicht-Fenster, 98
Layout-Auswahl-Fenster, 98
LicenceSessionInfo-Objekt, 79
Lizenz kaufen, 146
Lizenz-Konzept, 67
Lizenzanzeige, 141
Login, 54, 58, 134

Modrewrite, 83

Online-Auslieferung
    direkte, 7
    indirekte, 7

Payment-Server
    Installation, 62
PaymentClient, 58, 59
PaymentCustomer, 58
PaymentHandler, 49
PaymentHandlerInterface Schnittstellenbeschreibung, 50
PaymentHandlerInterface und PaymentHandler, 56
PDF-Verschlüsselung, 68

Relationen, *siehe* Tabellen
RMI-Client, 50, 62
Root-Url, 107
Root-Verzeichnis, 107

Schnittstelle
    Authoring-Tool, 99
    Payment-Komponente, 50
Schnittstellen, *siehe* DB-Schnittstellen
Security-Servlet, 79
Server-Plugin, 78
Session-ID, 67

Shop-Einstellungen
    ändern, 89, 114
Shop-Einstellungen-Fenster, 102
    Allgemein, 97
    Handbuch, 107
    Internet, 97
    Payment, 97
ShopInfo, 140
Sicherheitstechniken, 68
Socket-Kommunikation, 80
Suche, 146
Suchen, 137

Tabellen
    DDS, 19
    Payment-Handler, 26
Transaction, 55, 58
TransactionDoesNotExistException, 59
TransactionFailedException, 59
transactionSettled(), 52, 57

URI, 78

Wallet, 61
Warenkorb, 134
webshop.dbif, 28, 45
    Fehlerbehandlung, 42
    Klasse DB_Access, 28, 45
    Klasse DBMediator, 28, 45
    Paketstruktur, 46
WWW-Server
    Apache, 12

Zahlungsverfahren, 7, 49
    elektronische, 49, 61